NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EFFICIENT USE OF TELEPHONE SURVEY RESPONSE FACILITIES: A DECISION AID

by

Neale R. Ellis

September 1994

Thesis Advisor: D. P. Gaver

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Master's Thesis

4. TITLE AND SUBTITLE
EFFICIENT USE OF TELEPHONE SURVEY RESPONSE FACILITIES: A DECISION AID

5. FUNDING NUMBERS

6. AUTHOR(S)
Ellis, Neale R.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

12b. DISTRIBUTION CODE
A

13. ABSTRACT (Maximum 200 words)

This thesis develops an object-oriented simulation model of the Computer Aided Telephone Inquiry (CATI) system currently employed by the Defense Health Resources Study Center, which allows recipients of mailed survey questionnaires to respond to the mailed questionnaires via telephone. The simulation models system performance and the response arrival process as a transitory queuing system. The primary focus of this study is to develop a predictive decision aid for effective and efficient employment of the CATI system, while minimizing response attrition due to system overload. Sensitivity analysis is conducted to determine arrival rates which overload the system, mean service time effect on system capacity, and effects of various retry decision processes (i.e., the arrival process for respondents who fail to access the system because of system overload). Additionally, possible network optimizations designed to aid in the development of appropriate mailing strategies are discussed. As a predictive tool, the model appears to be quite accurate. Network optimization solutions for mailing strategies may achieve a significantly lower caller attrition rates than strategies which call for evenly distributed batch survey mailings.

14. SUBJECT TERMS	15. NUMBER OF PAGES
CATI, DHRSC, Simulation Model, Stochastic Process, Survey Response, Telephone Survey Response	112
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited.

Efficient Use Of Telephone Survey Response Facilities: A Decision Aid

by

Neale R. Ellis
Lieutenant, United States Navy
B. A., University of Virginia, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1994

Author: Neale R. Ellis
Neale R. Ellis

Approved By: D. P. Gaver
D. P. Gaver, Thesis Advisor

P. A. Jacobs
P. A. Jacobs, Second Reader

Peter Purdue
Peter Purdue, Chairman,
Department of Operations Research

ABSTRACT

This thesis develops an object-oriented simulation model of the Computer Aided Telephone Inquiry (CATI) system currently employed by the Defense Health Resources Study Center, which allows recipients of mailed survey questionnaires to respond to the mailed questionnaires via telephone. The simulation models system performance and the response arrival process as a transitory queuing system. The primary focus of this study is to develop a predictive decision aid for effective and efficient employment of the CATI system, while minimizing response attrition due to system overload. Sensitivity analysis is conducted to determine arrival rates which overload the system, mean service time effect on system capacity, and effects of various retry decision processes (i.e., the arrival process for respondents who fail to access the system because of system overload). Additionally, possible network optimizations designed to aid in the development of appropriate mailing strategies are discussed. As a predictive tool, the model appears to be quite accurate. Network optimization solutions for mailing strategies may achieve a significantly lower caller attrition rates than strategies which call for evenly distributed batch survey mailings.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this thesis may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

Additionally, a portion of the analysis conducted for this thesis was performed using *APL2/PC* and *AGSS*. The Naval Postgraduate School uses this program under a test agreement with IBM Research.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This thesis develops an object-oriented simulation model of the Computer Aided Telephone Inquiry (CATI) system currently employed by the Defense Health Resources Study Center, which allows persons to respond to mailed questionnaires via telephone. The program is written entirely in the object-oriented programming language MODSIM II. The simulation models system performance and the response arrival process as a transitory queuing system; such systems, in which all demand ultimately ceases, were discussed by Gaver, Lehoczky and Perlas (1975).

The Defense Health Resources Study Center (DHRSC) wishes to cost out the CATI system to prospective clients by maximizing total system utilization while constraining the probability that the system will overload (thus possibly reducing the number of people responding to a survey) to an acceptable degree. Providing an effective decision aid involves several key elements, the most important of which is developing a predictive model for survey response rates based on several important influencing variables such as survey length, survey distribution parameters, number of surveys mailed on a given day or days,...etc. The maximum number of people actually in the system at any one time is constrained by the number of telephone lines (currently 96). This thesis seeks to develop a decision aid for planning future usage of the system employing both mathematical and simulation methods. Future usage is anticipated to include the sequential, perhaps overlapping, service of different surveys.

The CATI system model is composed of three main elements: the system manager, the individual servers (phone lines), and the surveys (i.e. sets of questionnaires). The system manager routes (simulated) calls to individual servers (telephone lines) and keeps track of server status as well as system status. When an incoming call is received by the manager,

the manager first determines whether there are any servers currently idle. If there are available servers, the call (response) is routed to one of the idle servers and the time of the arrival and personal identification number of the caller are logged. If all lines tasked with conducting the survey are busy, the manager informs the caller, who must then decide if and when to attempt a recall.

The arrival process for respondents to a survey is based on conditional distributions fitted to empirical data from DHRSC Phase I. Arrival rates are non-stationary and vary depending on the size of a batch mailing of questioners and the day-of-week on which the questioners are mailed.

An examination of the effects of various levels of overload on the response percentage of a survey reveals that the system is remarkably insensitive to the effects of overload when the amount of overload (time in overload) is relatively small. The amount of caller attrition grows as the length of time that the system is in overload increases. Exploration into possible effects of the "Call-Back Server" strategy vs. different levels of customer call-back propensity suggests that the use of "Call-Back Servers" (servers that request callers to call back when the system is less busy) may not be an effective use of system resources. Sensitivity analysis in terms of system throughput for different survey (response) lengths (i.e. different location and shift parameters for the service time distribution parameters) indicates a plausible relationship between mean service time and system capacity (as mean service time decreases, system capacity increases, and vice versa). A comparison of model results to actual data from a later-phase survey, indicates that the model is surprisingly accurate as a predictor of future system performance. Alternative mailing strategies are discussed in terms of effectiveness using several approximate network linear/integer programming models. *Network solutions for mailing*

strategies may achieve a significantly lower caller attrition rates than strategies which call for evenly distributed batch survey mailings.

This model has been developed to be used, and to grow; it is not mature or complete. Its Object Oriented Design makes significant changes fairly simple. It can already handle several types of surveys running simultaneously. Future modifications might include features such as modeling system failure, or effects of scheduled maintenance and holidays, as well as improving the efficiency of the code and data structures.

I. INTRODUCTION

A. BACKGROUND

The Computer Aided Telephone Inquiry (CATI) system is a telephone survey response facility currently under the cognizance of the Defense Health Resources Study Center (DHRSC) working in conjunction with the Naval Postgraduate School Operations Research Department. It is a system designed to receive survey responses via a 96-line telephone system. The system is controlled by a one-server network of five computers based on the 486 CPU architecture.

System operation logic is as follows:

Operation: When a call comes into the system, it is automatically routed by the server to one of the 96 response lines, provided one of those lines is available, and response to the survey questionnaire is initiated. Each time a call enters the system (i.e. the system answers the telephone ring) it is given a Date/Time stamp regardless of whether the call results in a successful survey completion. If all lines are busy the caller receives a busy signal and the system does not register the call, making it difficult to keep accurate data on call response rates and, more significantly, possibly reducing the chance of the caller eventually responding to the survey. For this reason several lines (default = 3) are solely dedicated to receiving calls and requesting respondents to call back later when the system is not busy.[1] The logic behind this setup is that it is better for a caller to get through to the system, if only to be told to call back later, than for a caller to receive a busy signal. It is postulated that callers getting a 'call back' message will have a higher propensity to attempt a retry than those receiving a busy signal. Additionally, the computer cannot log

[1] Calls that are routed to a server tasked with delivering a call back message are currently not logged in and the Personal Identification Number (PIN) of the individual is not recorded.

the number of callers receiving a busy signal, making the loss of arrival data a near certainty. Each time the system reaches maximum capacity (including 'call back' lines), the manager will allocate three more lines to 'call back' duty. When system load drops below maximum capacity, servers are reallocated to 'Survey' duty. It can be seen that, carried to extremes, this would promote instability in the system.

At the present time the system is designed to respond to only one survey at a time. However, requirements for cost-effective use of the system indicates the need for a capability to handle multiple surveys simultaneously. The system is evolving rapidly toward this capability.

B. PROBLEM STATEMENT

DHRSC wishes to cost out the CATI system to prospective clients by maximizing total system utilization while constraining the probability that the system will overload (thus reducing the number of people responding to a survey) to an acceptable degree. A client wishes to receive as many completed questionnaires as possible, presumably as quickly as possible. Providing an effective decision aid involves several key elements, the most important of which is developing a predictive model for survey response rates based on several important influencing variables such as survey length, survey distribution parameters, number of surveys mailed on a given day or days,...etc. The maximum number of people actually in the system at any one time is constrained by the number of telephone lines (currently 96). This thesis seeks to develop a decision aid for planning future usage of the system employing both mathematical and simulation methods. Future usage is anticipated to include the sequential, perhaps overlapping, service of different surveys.

II. MODEL

A. SYSTEM

The CATI system model is composed of three main elements: the system manager, the individual servers (phone lines), and the surveys (i.e. sets of questionnaires). In this context, **surveys** refers to the mechanism which generates the arrival process, which will be discussed in following sections.

1. System Manager

The system manager routes (simulated) calls to individual servers (telephone lines) and keeps track of server status as well as system status. When an incoming call is received by the manager, the manager first determines whether there are any servers currently idle. If there are available servers, the call (response) is routed to one of the idle servers and the time of the arrival and personal identification number of the caller are logged. If all lines tasked with conducting the survey are busy[2], the manager informs the caller, who must then decide if and when to attempt a recall.[3]

---

[2] As a matter of procedure servers 94, 95, and 96 are reserved for the sole purpose of asking respondents to call back when the system is not as busy (thus only 93 servers are actually conducting surveys); when all lines are busy (94..96 inclusive), three additional lines are tasked with requesting a call back (now only 90 are conducting surveys). This continues until the system can receive all incoming calls. While this retasking is currently performed manually, the model assumes instantaneous response to an overload situation.

[3] There is very little data to support analysis of this behavioral decision process, however rationale for its use will be discussed later.

2. Server States

Servers have two possible states[4]; busy or idle. When busy, a server is in the process of conducting the survey questionnaire of an individual customer. When idle, the server is waiting for the manager to route another call.

B. ARRIVAL PROCESS

The basic model follows a four-tiered process to describe the arrival distribution of mailed surveys.

1. Propensity to Respond

Let $\mathbf{N}_i$ be the number of surveys mailed on day i. Let $\mathbf{p}_i$ be the probability that a single survey mailed on day i generates a call-in (regardless of whether it results in a completed survey). Then, if $\mathbf{R}_i$ is the total number of responses to result from a batch mailing on day i, $\mathbf{R}_i \sim \text{Binomial}(\mathbf{N}_i, \mathbf{p}_i)$.

2. Day of Response

Let $\mathbf{Q}_{ij}$ be the total number of responses (arrivals) received from a batch mailing on day i prior to day j. Let $\mathbf{d}_{w,(j-i)}$ be the conditional probability that a respondent, given that person is going to respond and has not responded prior to day j, will respond on day j (where w is the day of week on which mailing i occurred: i.e. Mon.,Tues.,Wed,...Fri.). Then, if $\mathbf{A}_{ij}$ is the number of responses from batch (mailed on day i) on day j, $\mathbf{A}_{ij} \sim \text{Binomial}(\mathbf{R}_i - \mathbf{Q}_{ij}, \mathbf{d}_{w,(j-i)})$.

---

[4] There is no claim that this state specification makes the system Markov. However, as will be discussed later, model insensitivity at high capacity might allow for the assumption of an exponential service time: which would make the system Markov.

3. Time of Response

Given A_{ij} arrivals on a single day from a batch mailed on day i, the hourly arrivals follow a Multinomial distribution $(A_{ij},c_1,c_2,...,c_{24})$ where c_h is the probability of responding during hour h. Individual arrivals are assumed to follow the behavior of the Poisson Process with mean = hourly arrivals. Although the structure of this model clearly indicates that individual arrivals are not independent because of finite population constraints, this model's value lies primarily with description of system behavior under **heavy loads** where the number of respondents involved are large enough to justify an independence assumption.

4. Success of Response

Given that a response is handled by the system, let **s** be the probability that the response results in a completed survey. Then for each response, successful completion is a Bernoulli trial with **s** as the probability of success.

C. SERVICE TIMES

Given that a response results in a completed survey, service times are IID Gamma(γ,α,β)[5] (three-parameter gamma distribution- Law and Kelton [Ref. 1:pp. 400-402]); otherwise, given that the response does not result in a completed survey, service times are IID Exponential(λ).

D. MODEL JUSTIFICATION

The primary focus of this study is to develop a useful predictive decision aid for employment of the CATI system. Limited available data (primarily because the system is

[5] In some cases; an exponential distribution with mean matched to the gamma mean, might prove adequate (see Footnote 4).

in its infancy, with only two major surveys serviced), makes it imperative that the study use the data as efficiently as possible. While conceptually the model was developed independently of the data, much of the structure of the model was influenced by the available data. For example; ideally the model should break the conditional day of arrival ($\mathbf{d}_{w,(j-i)}$) into two components. First, a distribution for time until receipt of a mailed survey, given the day on which it was mailed. Second, a distribution for the time to respond (arrive), given the day on which the survey is received. Because the available data does not allow for extraction of these separate distributions, they are treated as one distribution in the form of $\mathbf{d}_{w,(j-i)}$ (essentially multinomial in nature). While this, and other, concessions to the data might reduce the model's ability to capture the variability of the system/arrival process, the model appears to be usefully accurate as a predictive tool. As will be seen in later chapters, the low variability of the model makes this model more closely resemble an expected-value model, rather than an accurate descriptor of the confidence intervals for various system states. Capturing this variability should be a primary goal of further studies. The risk associated with a particular instrument and pattern of mailings can be assessed sequentially, and corrective action taken, but this process is not evaluated here.

1. Case-Specific Parameters

As will be discussed later, many of the input parameters to the simulation model are case-specific. Differences in the target population, time of year that the survey is conducted, survey length, human factor issues involved in the construction of the survey itself, and political climate are some of the possible influences on model parameters. This highlights the need for trial or pilot surveys to allow for appropriate estimation of these parameters.

Intuitively, however, the conditional day-of -response ($\mathbf{d}_{w,(j-i)}$) and time-of-response parameters should remain fairly constant (short of drastic improvements to the Postal Service or localized mailings, i.e. all mailing to the same time zone). This is not to say that these parameters can not be improved upon; only that they could be somewhat independent of specific scenarios. Strategies for model improvement (i.e. adaptation to new conditions) will also be discussed as recommendations.

III. DATA ANALYSIS

Data used for development of the model was based on the first DHRSC Survey conducted. This survey consisted of a total of 494,358 surveys mailed. Responses were monitored from January 24, 1994 through February 25, 1994. The following table indicates the mailing dates and quantity mailed.

Mailed Surveys	
Date	Quantity Mailed
24-Jan-94	24,000
25-Jan-94	25,500
26-Jan-94	38,700
27-Jan-94	39,000
28-Jan-94	40,000
31-Jan-94	52,884
1-Feb-94	38,000
2-Feb-94	90,000
3-Feb-94	54,612
4-Feb-94	90,829
7-Feb-94	833
TOTAL	494,358

Table 1

Each survey mailed was assigned a unique Personal Identification Number (PIN) from one to 494,358. The surveys were mailed (in batches) sequentially according to PIN.

Data extracted from the CATI system occurs in the following format:

Sample

Completion 0 = Not Completed 1=Completed	**Date**	**Time**	**Length of Call** (seconds)	**PIN**
0	19940124	1624	87	123
1	19940212	1059	312	268950

Table 2

Initial data analysis revealed significant time-of-day and day-of-week effects in system arrivals and, as expected, date of response is dependent on the date of mailing. Further analysis of the data with reference to the date of mailing indicate the surveys mailed on the same day of the week (regardless of which week) have similar times of survey response.

A. PARAMETERS

1. Propensity to Respond

Propensity to respond for a survey mailed on day i ($\hat{p}_i$), was obtained by taking the ratio of survey responses to total surveys mailed. A Chi-square Test was conducted to test credibility of the hypothesis that propensity was equivalent for all mailing days, with the test statistic,

$$\chi^2 = \sum_{i=1}^{k} \frac{[n_i - n\hat{p}]^2}{n\hat{p}} \qquad (1)$$

where n_i is the observed number of responses from batch mailing i

The test statistic was computed for k=11 and also for k=10 (excluding day 11 as a possible outlier).[6]

[6] Although day 11 appears to be dramatically different from the other days, the lack of control on the mailing and preparation process leaves some doubt as to its acceptability as a valid data point.

Mail #(i)	Date of Mailing	Respondents(n_i)	Total Qty Mailed(n)	p_i
1	24-Jan-94	5,533	24,000	.231
2	25-Jan-94	7,649	25,500	.300
3	26-Jan-94	10,211	38,700	.264
4	27-Jan-94	9,913	39,000	.254
5	28-Jan-94	10,060	40,000	.252
6	31-Jan-94	13,671	52,884	.259
7	1-Feb-94	9,537	38,000	.251
8	2-Feb-94	20,070	90,000	.223
9	3-Feb-94	13,140	54,612	.241
10	4-Feb-94	24,698	90,829	.272
11	7-Feb-94	573	833	.688
	Overall	125,055	494,358	**.253** ← $\hat{p}$

Table 3

Although Chi-Square rejects H_o in both cases, as noted by Law and Kelton [Ref. 1:pp. 380-382], "... if n is very large, then these tests will almost always reject H_o." Based on this observation, and the inherent variability/reliability problems of the postal system; a common $\hat{p}$ of 0.25 was chosen.[7]

2. Day of Response

The conditional probability $d_{w,(j-i)}$ that, given a respondent whose survey was mailed on day of week **w** has not called prior to day j, will respond on day j was obtained from an empirical density analysis by day of week (i.e. a separate conditional distribution for responses to surveys mailed on each day of the week [Monday ... Friday]), using the ratio of number of respondents on day j to (total number of responses - number of responses occurring prior to day j). All significant day-of-mailing effects occurred in the first partial week containing the mailing. Surprisingly, the following week (in which the bulk of total

[7] As a check to the impact of this assumption, the simulation model was altered to specify the individual p_i, although output was somewhat different, system overload prediction was nearly identical (less than 5% arrival difference in any one day), indicating that no significant information was lost using a single **p** value.

responses occur) behaved similarly regardless of day-of-week of mailing. The following scatter plot displays the ratio of responses on day j to the total number of responses that have not occurred prior to day j.[8] For notation purposes, week Zero refers to the initial days of response prior to the first Monday following the day of mailing. Note that week zero is of different length depending on the day of week of the mailing. Thus while day 13 on this scatter plot indicates the thirteenth day since the Monday batch mailings, it represents only the ninth day since the Friday batch mailings. Week One through week Five refer to the following full weeks of arrivals (Monday-Sunday). When looking at this plot it is important to note that 74 % of all arrivals have occurred by day 13 (end of week one) and 98 % of all arrivals have occurred by day 20 (end of week two). For purposes of using these data to obtain parameter estimates for daily arrivals, several points must be addressed. First, with the exception of week zero, residual response ratios were nearly identical, regardless of the day of week of the mailing. Thus, residual response dependency on the day of week of the mailing appeared only to be significant in week zero. Second, this data set is truncated at the 32 day mark; however the model allows for variable system coverage which might exceed 32 days. After day 32, the number of calls expected is extremely small[9], but there is still a positive probability of generating an arrival. As the available data is truncated past day 32, the small number of calls that might have occurred after day 32 could impact the residual ratios after week one due to changes in the relatively small number of remaining responses. This fact makes using the empirical ratios after week one suspect. For these reasons, parameters for the conditional

[8] This ratio will also be referred to as a residual response ratio.

[9] For example: the number of calls arriving from 40,000 questionnaires mailed on 28 Jan,1994, resulted in 7294 arrivals during week one, 1660 arrivals during week two, but only 40 arrivals during week four.

probability $\mathbf{d}_{w,(j-i)}$ for week zero are taken from week zero data (specific to day of week of mailing) and all subsequent weeks are fitted to week-one data.[10]

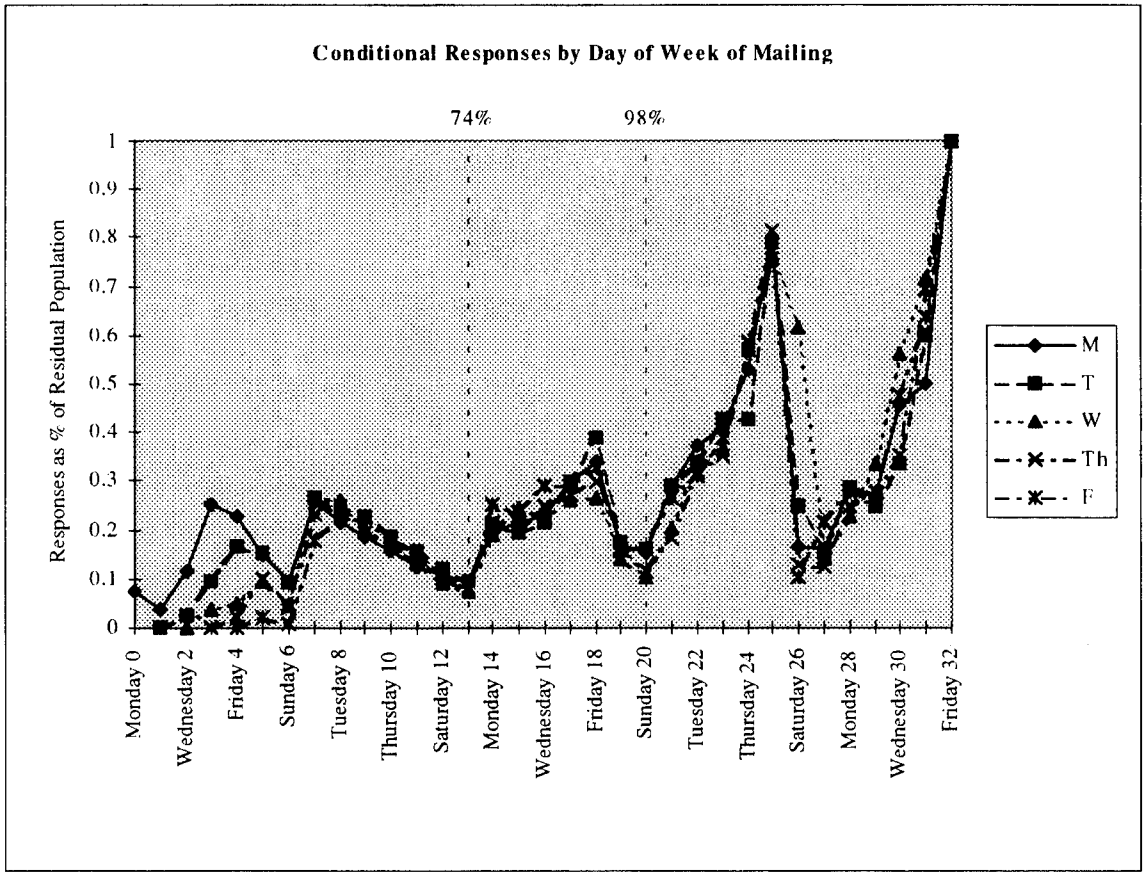

Figure 1-*Note: First Data Point for Each Line Indicates the Day on Which That Mailing Occurred.*

The following table represents conditional response probabilities used by the simulation model:

[10] Since the model's value is derived primarily from looking at the system at high capacity, the loss of precision in regards to parameters after week one becomes a fairly insignificant problem in terms of load prediction as the number of arrivals is so small. See Results chapter Section B.

	Week 0					Week >= 1
Day-of-Week	Mon-Mailing	Tue-Mailing	Wed-Mailing	Thu-Mailing	Fri-Mailing	All
Mon	0	0	0	0	0	0.3
Tue	0.05	0	0	0	0	0.27
Wed	0.11	0.05	0	0	0	0.24
Thu	0.3	0.1	0.04	0	0	0.21
Fri	0.27	0.18	0.05	0.02	0	0.18
Sat	0.24	0.15	0.09	0.1	0.02	0.15
Sun	0.21	0.09	0.04	0.04	0.01	0.12

Table 4 - Conditional Response Probabilities

3. Time of Response

Not wishing to 'over-fit' the data, the decision was made to use a multinomial distribution instead of an empirical density (simplicity of implementation was also a factor) for the purpose of describing time-of-response that a call will arrive, given that it arrives on a specific day. The Multinomial distribution $(\mathbf{A}_{ij},\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_{24})$ is fitted by taking the empirical ratios of total hourly arrivals to total arrivals. Times for arrivals within each hour are assumed to be independent uniform (unordered) over the hour.

The following bar graph represents the pdf of the simulation time-of-response Multinomial.

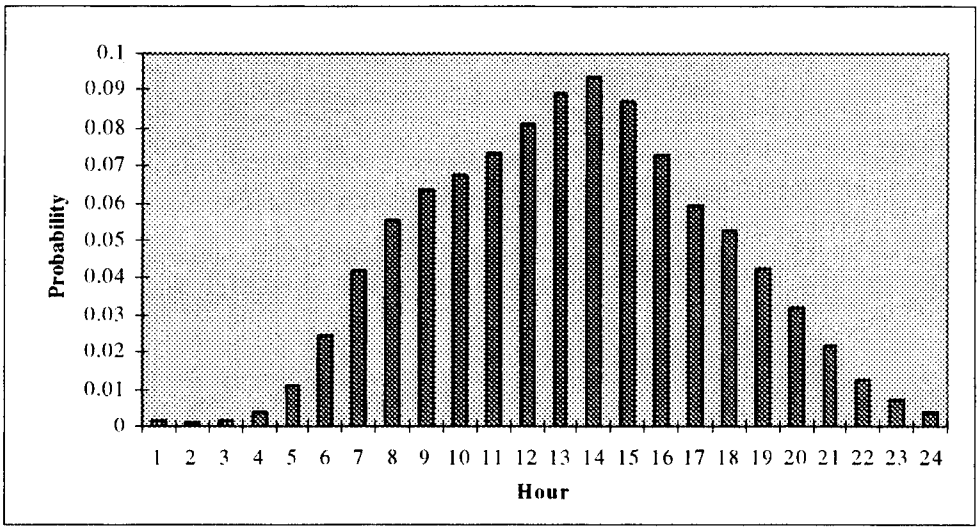

Figure 2

The estimated multinomial probabilities are listed in Appendix A, Implementation Module Survey source code, Multinomial Method.

4. Success of Response

Success-of-response probability **s** (the probability that a call that started a survey will successfully complete that survey) is estimated by the ratio of calls resulting in a completed survey to the total number of calls received ($\hat{\mathbf{s}} = \mathbf{0.835}$).

B. SERVICE TIMES

The distribution of service times, given that the call results in successful completion of a survey, has been successfully fitted by a three parameter Gamma density ($\hat{\gamma} = \mathbf{113.954}(\text{sec}), \hat{\alpha} = \mathbf{5.217}, \hat{\beta} = \mathbf{43.109}$) with density:

$$f(x) = \left\{ \frac{\beta^{-\alpha}(x-\gamma)^{\alpha-1} e^{-(x-\gamma)/\beta}}{\Gamma(\alpha)} \quad \text{if } x > \gamma, \text{ otherwise } f(x) = 0 \right\} \qquad (2)$$

The parameters for the shifted gamma distribution were obtained using maximum product of spacing (MPS) estimation as described by Law and Kelton [Ref. 1:pp. 400-402] based on a sample size of 5142. Below is a comparison of the CDF of the fitted Gamma to the (successful completion) service time data.

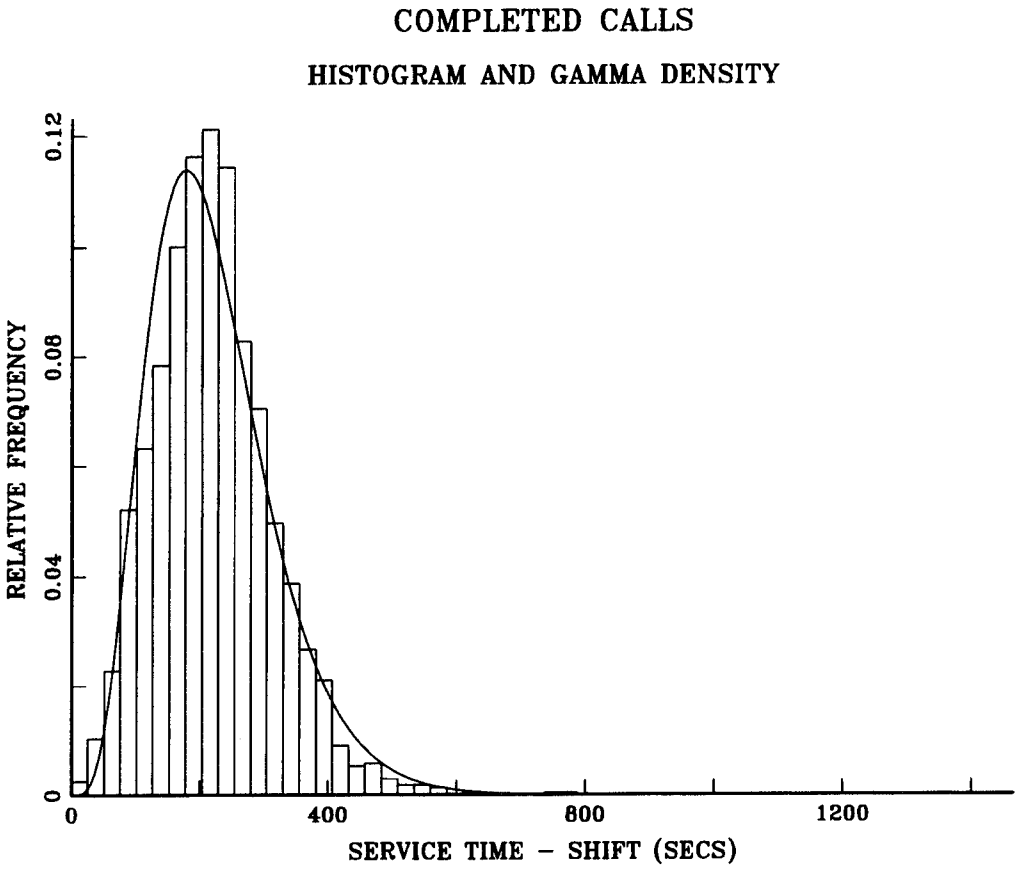

Figure 3

The distribution of service times, given that the call **does not** result in successful completion of a survey (for callers who were connected to a 'survey' server) was fitted to an exponential density (mean=177 sec). The parameter for the exponential distribution was obtained using maximum likelihood estimation.

The following is an overlay of an Exponential pdf (λ=177) and a histogram of the unsuccessful call service times.

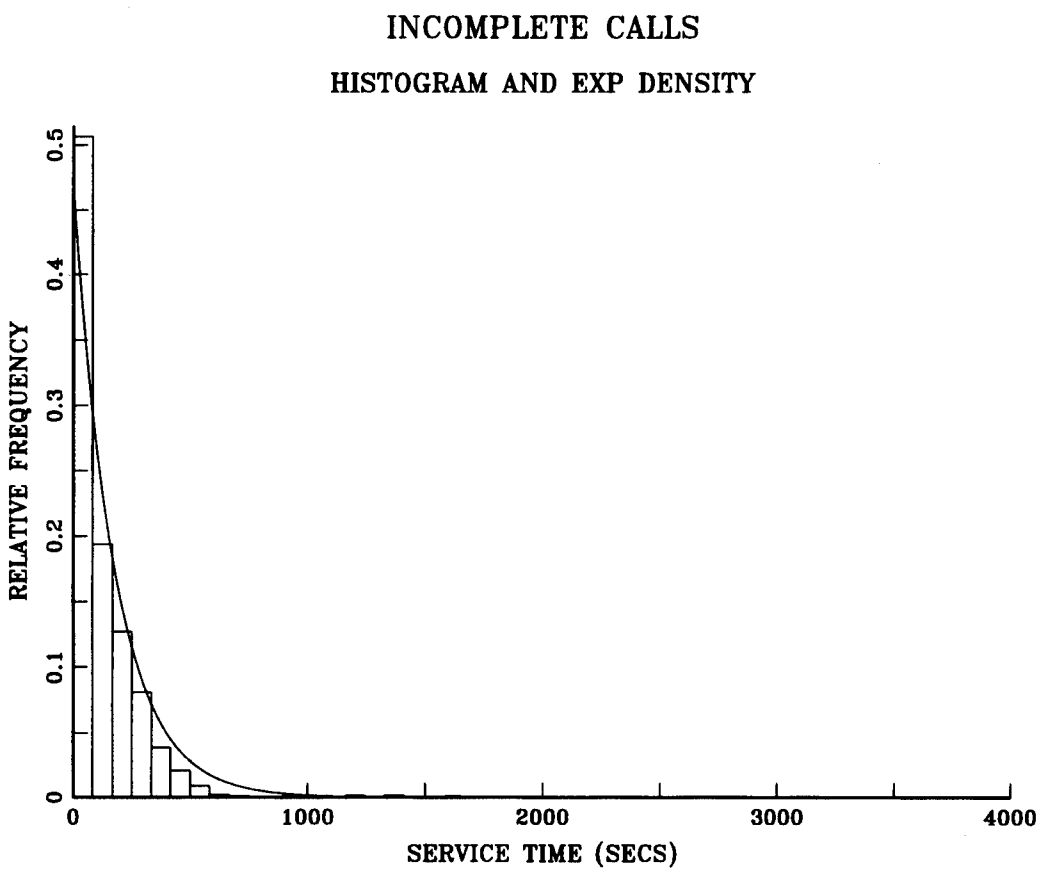

Figure 4

The distribution for service times, given the call **does not result** in a successful completion of a survey, are Exponential (mean=177 sec).

IV. MODEL BEHAVIOR

While details of the inter-workings of the model will be discussed in later chapters, the basic functionality of the model can be easily summed up by a block flow diagram (Gaver and Jacobs[Ref. 2]).

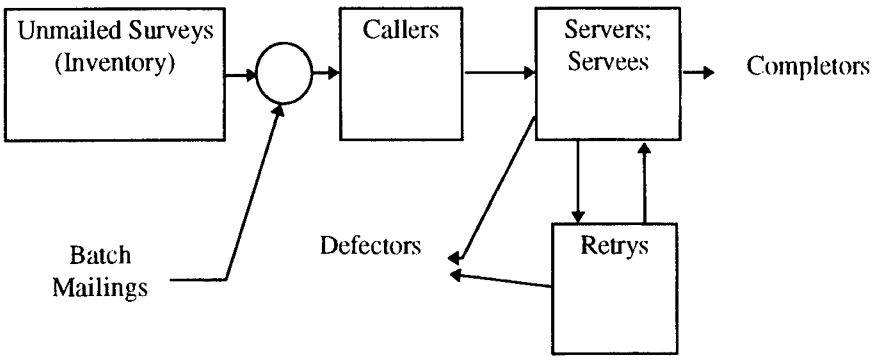

Figure 5

This chapter discusses various aspects of model behavior using multiple simulation runs. The following data were generated from the model using a scenario of five mailing days in one week (Monday through Friday). 100,000 surveys were "mailed" on each of the mailing days for a total of 500,000 surveys. Input parameters to the model include:

- 96 servers/Default call back server strategy.
- Service Time distributions as previously discussed for DHRSC Survey.

•Retry (Call Back) probability and wait time distribution are equivalent for both busy signal initiated call backs and "Call-Back server" initiated call backs- 0.5 and EXP(mean=30 min) respectively.

A. ARRIVALS (BY DAY)

The following is a graph depicting the average number of arrivals by day for the scenario. The line representing Total Arrivals refers to the total number of calls attempting to enter the system on a day. In terms of the diagram, Total Arrivals is the sum of Retries and Callers. Original Arrivals is equivalent to Callers and refers to the total number of individuals who attempt to access the system. Enter System refers to the total number of people who actually get into the system and are routed to a server for the purpose of conducting a survey. It is equivalent to the sum of Completors and the portion of Defectors who were in the process of responding to a survey (i.e. those starting a survey, but leaving before being completely served by a telephone "survey" server).

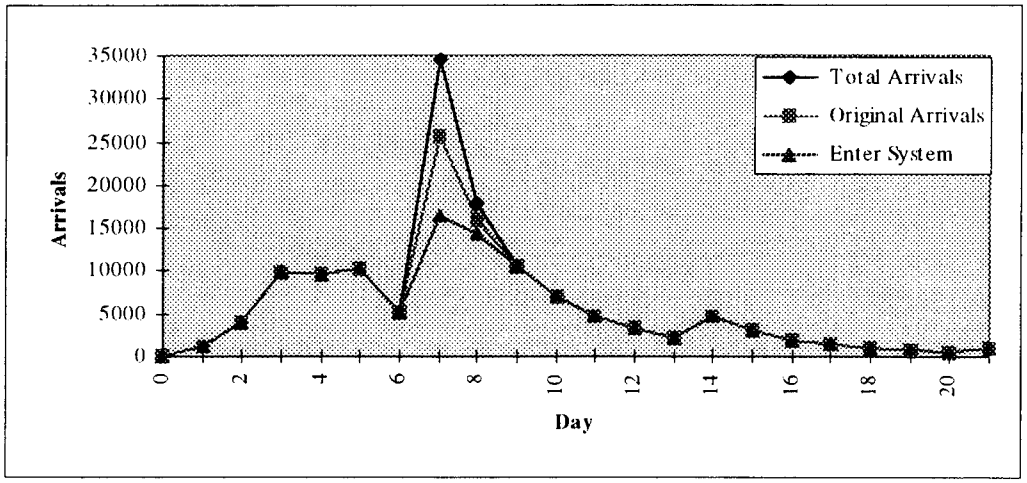

Figure 6

Clearly the three totals are essentially the same number when system resources are sufficient to handle the number of callers attempting to access the system, because no one is being sent into retry status. However, when system resources are insufficient, the difference may become dramatic. On the seventh simulation day, the number of individuals attempting to access the system was 25,520, but only 16,407 of those individuals were connected to a server responsible for conducting the survey. This implies that at least 9,113 potential responses were blocked and lost forever. The following chart displays the number of callers in a Retry state as well as the number of busy servers for the sixth through the tenth days of simulation:

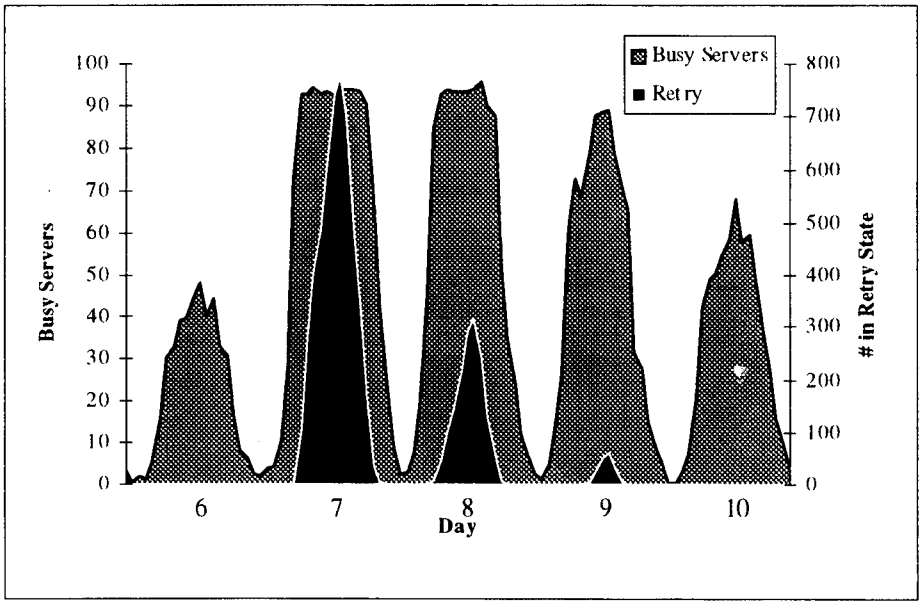

Figure 7

Note that the Day Axis marks continuous time starting at 12 AM on the morning of the sixth day and ending at Midnight on the evening of the tenth day. The day labels are centered at 12 PM of each day. On the seventh and eighth days (days of largest overload) the system is working at full capacity for most of the day (note the large plateau in number

of busy servers), forcing a large number of callers into Retry status. Hence, the length of time that the system is operating at full capacity is also related to the number of callers forced into Retry status, and, of course, the number of potential responses lost. It is desirable to minimize this number where possible so as to obtain as many completed responses as possible, given the desirability of finishing the survey soon.

B. CALL BACK SERVERS

The following graph is a representation of the Survey Server/Call-Back Server occupancy distribution over the sixth through the tenth simulation days.

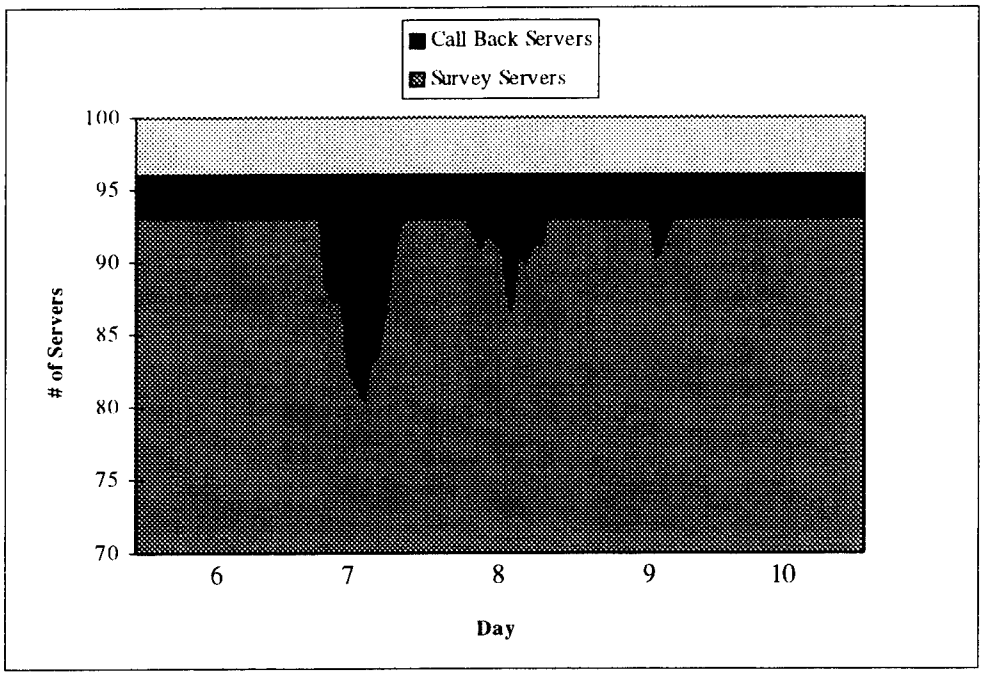

Figure 8

The high level of overload experienced on day seven causes system logic to allocate up to 16 of 96 servers to call-back status leaving only 80 servers to actually conduct surveys. This means that, at the systems busiest period, almost 17 % of the system's effective

survey-handling capability is lost. It will be shown in later chapters that the current Call-Back Server strategy is inefficient even if using the Call Back Server yields a much higher probability of Retry.

C. ATTRITION MEASUREMENT

While acceptable levels of response attrition is a subjective matter for the decision maker; scale (not surprisingly) is a critical factor in determining how one views the impact of overload conditions on a survey's cost-effectiveness. This study takes the point of view that attrition should be measured as a fraction of the expected number of responses, given unlimited system resources (i.e. an infinite number of servers) are available. With this in mind, it becomes important to distinguish between losses in the context of all survey responses and losses in terms of the expected number of responses in a given day. From the standpoint of the sponsor of an individual survey, examining the losses in proportion to all potential survey responses seems the more useful of the two approaches. To date the CATI system has been utilized by a single sponsor with one basic survey. In such a case, the length of time allocated on the system for a specific survey is not a major issue, allowing the sponsor to spread out batch mailings sufficiently to effectively prevent system overload. Using the previously mentioned 500K survey scenario as an example of a single survey with time constraints, one can see that although overload levels on the seventh day resulted in 36 % attrition in potential responses for that day (second criteria), the overall losses were fairly insignificant (7.3 % of potential responses). In the context of total response rate (which for this model was assumed to be 25%) the total response percentage was reduced by only 2.2 % (losses as a percentage of total mailings vs. potential responses).

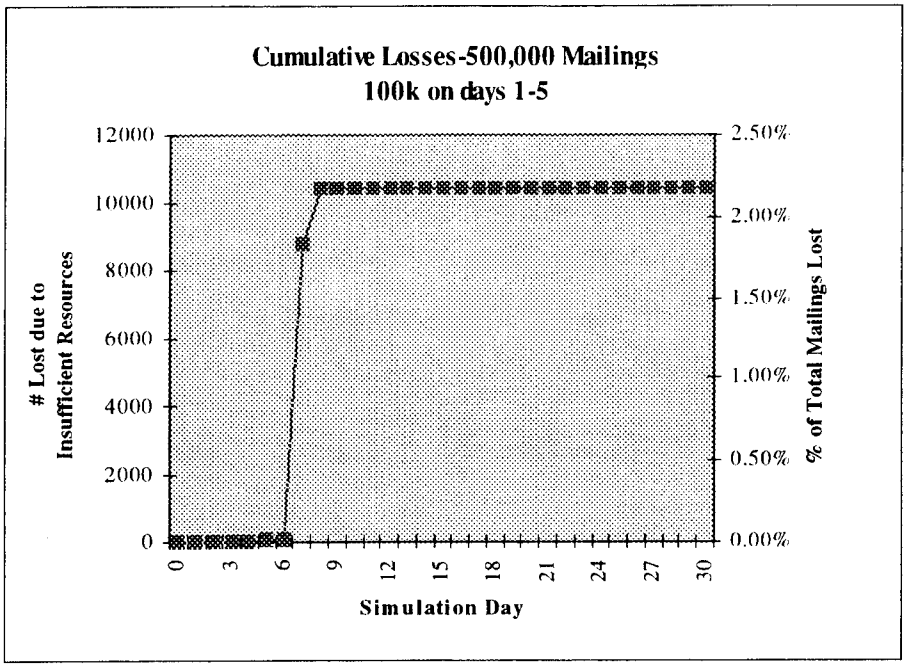

Figure 9

A 22.8 % response rate versus 25 %, might seem reasonable given the constraints of the scenario. However, from the standpoint of the system administrator interested in optimizing system utilization over time (i.e. maximizing utility/profit of the system by making it available for use by various agencies with possibly various types of surveys), the rate of loss, expressed as a proportion of the potential responses on a day, might be a more conservative criteria, as continuous periods of overload quickly raise the level of response attrition (note that almost all of the attrition in the previous scenario occurred on day seven). The above measure can be used both as a diagnostic and a criterion: one approach could be to minimize the maximum expected number of losses on each day of a survey's response.

V. RESULTS AND VALIDATION

This section will focus on five main areas. First, an examination of the effects of various levels of overload on the response percentage of a survey. Second, in conjunction with the first issue, exploration into possible effects of the "Call-Back Server" strategy vs. different levels of customer call-back propensity. Third, sensitivity analysis in terms of system throughput for different survey (response) lengths (i.e. different location and shift parameters for the service time Gamma Distribution and different means for the Exponential). Fourth, a discussion of the model as a predictor for future system performance. Fifth, alternative mailing strategies will be discussed in terms of effectiveness using several approximate network optimization models.

A. POTENTIAL RESPONSE VS RESPONSE UNDER OVERLOAD CONDITIONS

The following data are based on ten simulation runs with the following parameters:

- 96 servers using default call-back server procedures.[11]
- The probability that a person attempts a recall (either because of a computer message or busy signal) is 0.15.
- The amount of time before a call back is attempted is exponentially distributed with mean = 30 minutes (computer message or busy signal).

In the following tables and graphs all values computed are based on mean values for a single simulation day (on which overload occurred), with column definitions as follows: **Total Arrivals** refers to the total number of calls which attempted to enter the system (***includes recalls*** either from a 'call back' message or busy signal). **Original Arrivals**

[11] See Footnote 1.

refers to the total number of persons who attempted to access the system (***excludes all recalls***). **Survey Arrivals** refers to the number of callers who actually were routed to a server recording the survey questionnaire response.[12] **Lost** is the difference between Original Arrivals and Survey Arrivals (essentially the number of potential survey responses lost). **Daily%Lost** is the ratio of Lost to Original Arrivals.

Scenario #	Total Arrivals	Original Arrivals	Survey Arrivals	Lost	Daily%Lost
1	7679	7679	7673	6	0.08%
2	7780	7778	7767	11	0.14%
3	7795	7793	7783	10	0.13%
4	7790	7789	7782	7	0.09%
5	7876	7874	7850	24	0.30%
6	8092	8092	8084	8	0.10%
7	8393	8383	8332	51	0.61%
8	8974	8957	8838	119	1.33%
9	9592	9543	9298	245	2.57%
10	11583	11414	10533	881	7.72%
11	13157	12882	11277	1605	12.46%
12	14933	14455	11769	2686	18.58%
13	17045	16360	12470	3890	23.78%
14	18593	17725	12707	5018	28.31%
15	20401	19305	13032	6273	32.49%

Table 5

The previous table was constructed using the mean values (across ten replications for each scenario) for the peak overload day in the given scenario. All scenarios were identical except for the number of surveys mailed (and therefore the number attempting to access the system in a given simulation day)[13]. Table 4 reveals that, up to a point, achieving overload levels is not necessarily bad, in the sense that a small

[12] While an important survey design issue, this study does not focus on methods for improving the probability of successfully completing a survey, given one accesses a 'survey' server. For this reason, the number of people who have the opportunity to respond is used as a sufficient criteria of effectiveness

[13] The mailing pattern is not an issue in this case because the analysis is focusing on a single response day. Arrival distributions are identical, regardless of the day (only the number of arrivals differ). For all scenarios, batch mailings were mailed on a single day (Monday).

fraction/percentage of potential responses are lost. Although the system reached an initial overload state with 7,679 callers attempting to respond, significant losses[14] (>1%) in potential responses do not occur until an Original Arrival number of 8,957. This yields leeway on the order of 1000 arrivals per day, in terms of planning system resource allocation. Note also that, as the percentage of losses increases, so does system throughput. This is explained by the fact that as persons attempt to recall they are more likely to call at a time when the system has available servers (in other words; idle gaps in server usage are filled later, and the system becomes more efficient in terms of throughput).

B. IMPACT OF CALL-BACK PROBABILITY AND CALL-BACK WAIT TIMES

While the previous case examined overload conditions, treating the events of a caller receiving a call-back message and a caller receiving a busy signal as having equivalent effects(in both cases the probability of attempting a Retry and the time until Retry were (~Bernoulli(0.15),~Exponential(30min.) respectively), the following tables and graphs display varying levels of both these parameters.

[14] Of course what is significant depends heavily on the needs of the decision maker. 1% seems fairly conservative.

MSG(.15,30min) Busy(.15,30min) Series 1		MSG(.5,60min) Busy(.15,30min) Series 2		MSG(.5,60min) Busy(.5,10min) Series 3		MSG(.5,240min) Busy(.15,30min) Series 4		MSG(.9,30min) Busy(.05,20min) Series 5		MSG(.15,30min) Busy(.9,10min) Series 6	
Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost
7679	0.08%	7679	0.09%	7679	0.09%	7679	0.09%	7679	0.00%	7679	0.08%
7780	0.00%	7782	0.14%	7782	0.14%	7782	0.04%	7782	0.08%	7778	0.14%
7793	0.09%	7792	0.08%	7792	0.08%	7790	0.13%	7790	0.03%	7793	0.10%
7783	0.09%	7783	0.09%	7783	0.06%	7783	0.09%	7787	0.10%	7790	0.09%
7878	0.18%	7877	0.23%	7878	0.23%	7877	0.27%	7877	0.15%	7877	0.28%
8094	0.12%	8098	0.27%	8102	0.26%	8100	0.25%	8101	0.14%	8094	0.15%
8412	0.62%	8410	0.40%	8407	0.33%	8407	0.36%	8409	0.23%	8383	0.61%
8983	1.02%	8995	0.81%	8991	0.86%	8991	0.83%	8991	0.65%	8965	1.18%
9536	2.00%	9536	1.97%	9535	1.88%	9533	1.78%	9538	1.86%	9543	2.43%
11417	6.69%	11418	6.72%	11405	6.30%	11418	5.99%	11412	6.06%	11422	7.46%
12872	12.20%	12878	12.19%	12845	10.87%	12868	10.53%	12821	10.83%	12868	12.05%
14448	18.47%	14458	18.05%	14426	17.95%	14458	16.50%	14443	17.49%	14432	18.31%
16263	23.65%	16263	23.18%	16265	23.01%	16311	21.53%	16239	23.36%	16242	23.61%
17786	28.80%	17786	28.22%	17800	28.48%	17812	26.42%	17824	29.30%	17835	28.95%
19169	32.67%	19172	32.33%	19117	32.65%	19180	30.26%	19070	32.74%	19119	32.98%

Table 6

The data in this table was obtain by running 15 scenarios (identical to the previous section scenarios) for each listed variation of call-back parameters. As before, the data represents the simulation day of highest overload (the same day for all scenarios).[15] The first row of the table defines the call-back parameters. MSG(.15,30min)/Busy(.15,10min) specifies that the trial was conducted giving persons who receive a call-back message a 0.15 probability of attempting a recall, with a wait time until recall which is exponentially distributed with a mean of 30 minutes. Additionally, persons who receive a busy signal are assigned a 0.15 probability of attempting a recall with a wait time until recall which is Exponentially distributed with a mean of ten minutes. There is no limit on the number of times a person may recall, however, his probability of attempting a Retry at least **n** times is

[15] See Footnote 13.

p^n , where p is the probability of attempting a recall, given either busy signals or call-back messages on the retries (i.e. there is no increasing impatience modeled). A question may arise as to whether persons who call back on the next day (and therefore are not reflected in this table), impact the results. Given the structure of the implemented call-back logic, the number of persons who waited until the next day to retry was extremely small and had no significant impact on the results.[16]

Graphically, at acceptable levels of attrition, there is virtually no difference across treatments and even at relatively high levels of overload (>15 % attrition) there appears to be no significant advantage for any of the depicted situations.

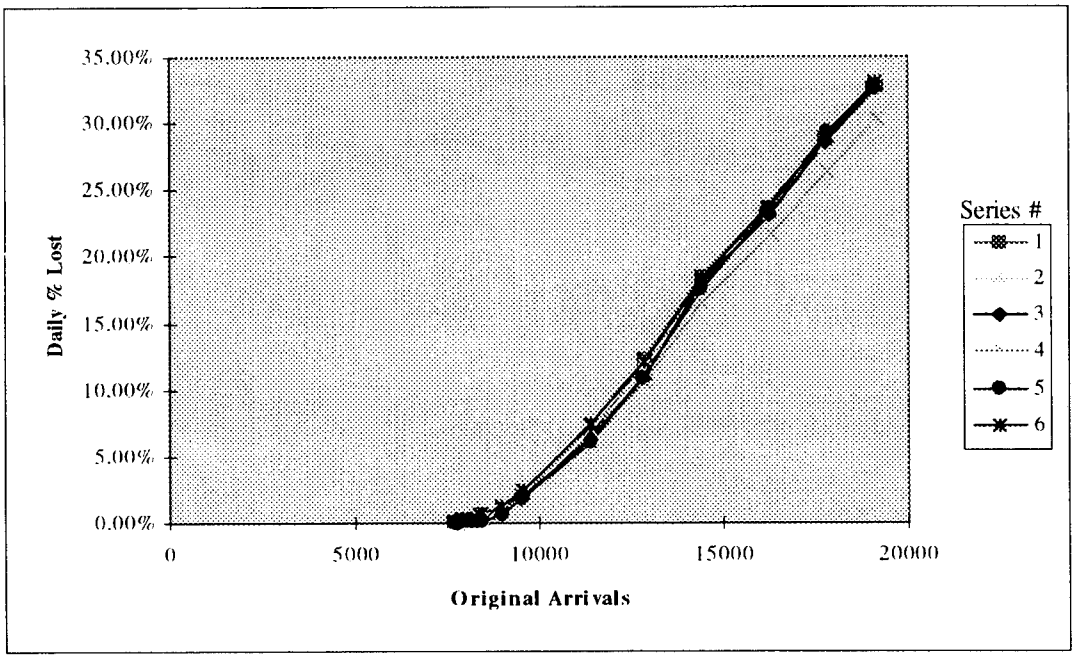

Figure 10

[16] This would not necessarily be true if the mean time to retry was increased sufficiently (i.e., large enough to cause first time retries to occur sometime after the current day - the mean wait time would probably need to be greater than eight hours to show significant numbers of next-day retries).

As an alternative to the use of the Call Back server strategy, the model was run utilizing all 96 servers strictly as survey conductors (i.e. no call back servers; all call backs are the result of a busy signal).

Busy(.001,30min) Series 7		Busy(.15,30min) Series 8		Busy(.5,30min) Series 9		Busy(.9,30min) Series 10	
Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost	Original Arrivals	Daily %Lost
7679	0.01%	7679	0.01%	7679	0.00%	7679	0.00%
7778	0.01%	7778	0.00%	7778	0.00%	7778	0.00%
7793	0.08%	7793	0.05%	7793	0.05%	7793	0.00%
7790	0.01%	7790	0.01%	7790	0.01%	7790	0.00%
7877	0.17%	7877	0.13%	7877	0.10%	7877	0.04%
8095	0.11%	8094	0.11%	8095	0.09%	8095	0.02%
8383	0.42%	8383	0.38%	8402	0.25%	8386	0.07%
8960	0.90%	8964	0.55%	8972	0.56%	8974	0.16%
9537	2.11%	9538	2.08%	9540	1.45%	9534	0.36%
11390	6.21%	11410	6.26%	11412	5.45%	11406	2.28%
12919	10.98%	12912	10.44%	12878	9.64%	12804	5.44%
14530	17.46%	14503	16.61%	14454	15.31%	14476	10.23%
16281	21.69%	16289	21.00%	16255	20.64%	16080	14.23%
17707	26.00%	17710	25.56%	17796	24.96%	17974	19.65%
19357	29.64%	19326	29.75%	19170	29.07%	19275	22.92%

Table 7

A comparison of Table 6 to Table 5 reveals that in almost all cases the system achieved lower attrition rates when no call-back server strategy was employed (especially if a high call-back probability exists). Of special note is the fact that the system achieves lower attrition rates (with no call-back servers) even when the probability of call back is near zero (.001-series 7). This suggests that dedicating all lines to survey service may be the more efficient mode of operation. Plotting series 7-10 with the previous data (series 1-6) yields the following:

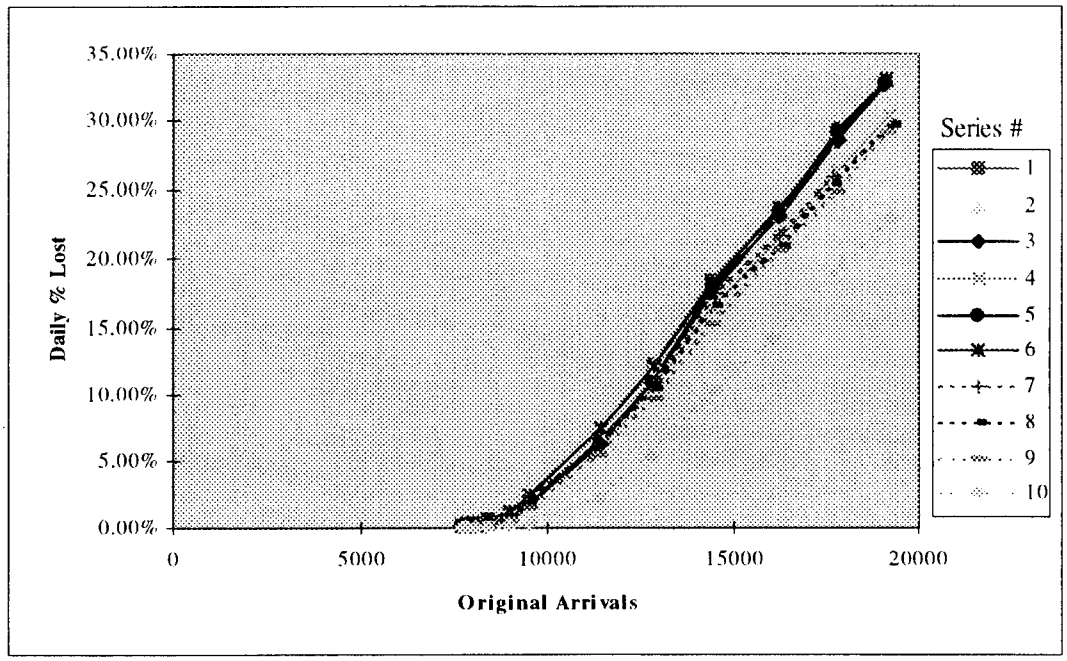

Figure 11

From the graph one can see that series seven through 10 (no Call-Back Servers) are at least as effective in reducing attrition as any of the scenarios utilizing Call-Back Servers. Note that increasing call-back probability to 0.9 in the case of a busy signal (series 10) appears to create a dramatic improvement, even at low levels of overload.

C. SERVICE TIME SENSITIVITY

The following graph is the result of system sensitivity to survey length (expected service times per call).

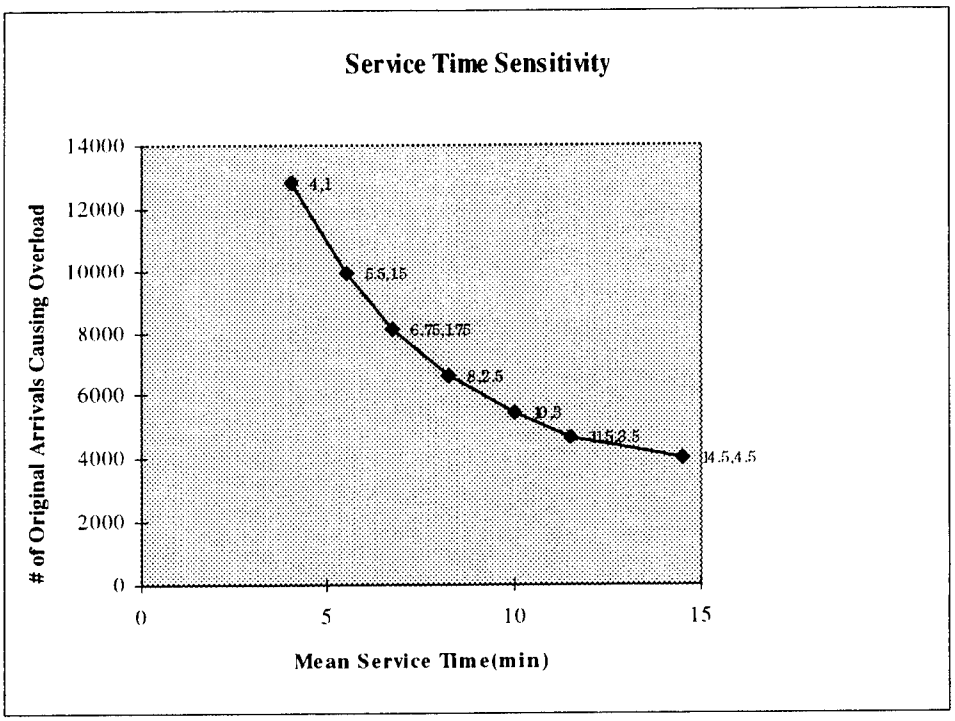

Figure 12

The simulation runs that generated this data used the default Call-Back Server strategy. The probability of call back and mean time for call back were 0.5 and 30 minutes respectively (same for both busy signals and message recipients). The text next to each of the data points lists the mean service time, and minimum service time (shift parameter) used in the simulation runs which created that data point (Alpha for the three parameter Gamma remained a constant 5.2 where β = [mean service time-minimum service time]/α). Overload conditions were defined as the minimum number of original arrivals in a simulation day that generated at least one instance of retry (in all 10 replications) as a result of a busy signal (i.e., all servers were busy when a new call arrived).[17] One pleasant

[17] For example: if one simulation run resulted in overload at 8,300 arrivals, but the other nine simulation runs did not, 8,300 arrivals would not have been used as an overload condition.

and desirable dividend: the model appears to be relatively insensitive to the form of service time distribution. The system behaves similarly at high capacity regardless of whether service times are sampled from a gamma or an exponential distribution.[18] The driving factor appears to be the mean service time in both cases. While this property will not be addressed directly by this thesis, it suggests the possibility of modeling the system as Markovian in nature.

D. ACCURACY AS A PREDICTIVE TOOL

Perhaps the greatest value of this model lies in its potential to predict system behavior in various scenarios. As a test and partial validation of the model, mailing strategy data were obtained for the second major survey to be conducted on the CATI (Phase III). The date and quantity mailed for each batch of surveys mailed in Phase III was obtained and used for input to the model. One of the four slave computers was out of commission for the second survey, reducing the number of available lines from 96 to 72. Subsequently, the number of servers in the system simulation model was reduced to 72, to parallel this reality. All other input parameters remained consistent with the results of initial data analysis discussed in Chapter III. The mailing strategy for Phase III was as follows:

[18] The data which supports this claim is not presented in this thesis.

Day-of-Week	Date Mailed	Quantity Mailed
Fri	3-Jun-94	23921
Mon	6-Jun-94	9995
Tue	7-Jun-94	24937
Wed	8-Jun-94	25030
Thu	9-Jun-94	14957
Mon	13-Jun-94	24980
Tue	14-Jun-94	24869
Wed	15-Jun-94	24882
Thu	16-Jun-94	14249
Mon	20-Jun-94	35639
Tue	21-Jun-94	25086
Wed	22-Jun-94	24992
Thu	23-Jun-94	25013
Mon	27-Jun-94	25507
Tue	28-Jun-94	24995
Wed	29-Jun-94	24995
Mon	5-Jul-94	29816
Total		**403863**

Table 8

Note that the survey batch mailings were spread out thinly over a period of 32 days with 17 distinct batch mailings to reduce the probability of system overload. The following graph is a comparison of the actual arrival data versus output generated from the simulation model. All parameters used in the simulation were identical to those discussed in the Data Analysis chapter.

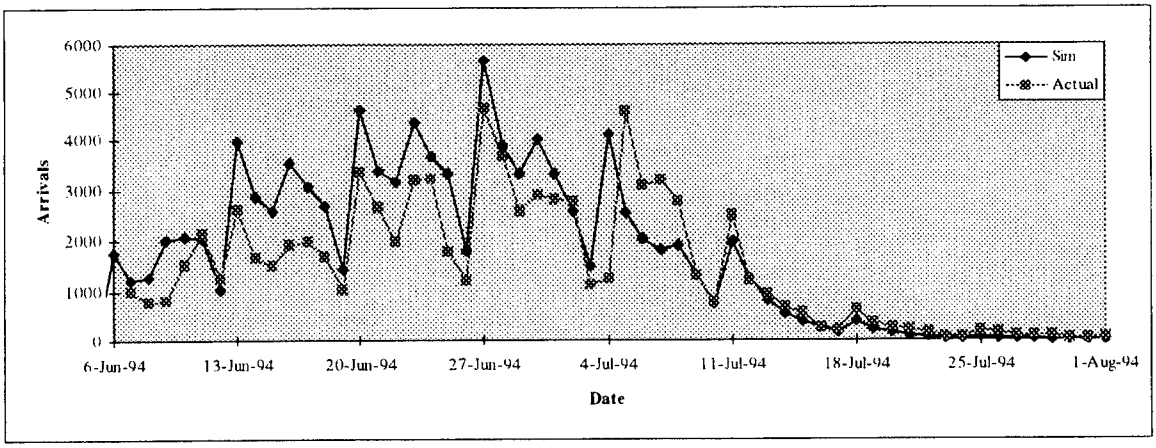

Figure 13

The mailing strategy was successful as there were no overload periods during Phase III. The model successfully predicted this event.

Clearly the patterns of arrivals are similar, but the data indicates that the model tends to over-estimate the numbers of arrivals. Additionally, there appears to be a pattern shift around 4-Jul-94. Explanations for both of these dissimilarities are intuitive. Over-estimation suggests that the overall propensity to respond was lower than the 0.25 which occurred in the first survey and was chosen as the input parameter value used by the simulation model. This, in fact, turned out to be true. Phase III propensity to respond via CATI was 0.206. Note that such a propensity can be estimated on-line from early data and used to modify a mailing strategy.

Propensity to respond could be dramatically influenced by many factors. The differences between the first DHRSC survey and the second were seemingly minimal yet the propensity to respond was somewhat lower for the second survey. Two factors appear to be likely candidates for the cause of this drop: timing and authenticity. The first survey was conducted from the end of January to the end of February 1994, while the second survey was conducted through June and July of that same year. Not only are the

summer months the peak period for Permanent Changes of Station (PCS) for the military community (the primary target population for these surveys),[19] but are also the primary vacation months in the United States. This might increase the likelihood of a mailed survey never reaching the desired person, or of reaching one such at a time when he/she has more important things with which to deal. Additionally, while the first set of surveys all bore the signature of the Assistant Secretary of Defense (Health Affairs), no signature was placed on the second set of surveys. It is possible that recipients of the second survey mailings either viewed it to be of lesser importance than did those who received the first survey, or they might even have suspected fraud (Iversen: Ref. 3).

The shift in arrival patterns on July 4th is expected for several reasons. First, July 4th fell on a Monday in 1994, which means that for most people (especially Government employees) Monday was still a part of the weekend. Second, because July 4th is a national holiday, there was no mail delivered on that particular Monday, delaying (and possibly reducing) a number of potential callers. Third, a July 4th holiday that ends on a Monday is a logical end to a vacation period. It is very possible that people who would have responded much earlier did not respond until July 5th because that was when they were able to catch up on their mail backlog. These three reasons, combined, most likely account for both the shift on July 4th, and also the slightly higher return rate immediately following the holiday weekend. Since the model does not take holidays into account, it fails to capture this exception. Although beyond the scope of this study, it would not be difficult to modify the model to account for holidays.[20]

Highlighting the importance of establishing a realistic propensity-to-respond parameter, assume that one was aware of this (possibly seasonal) drop (either through

---

[19] This is likely true of the civilian population also as the summer months are more convenient in terms of relocation for families with school age children.

[20] One possibility could be to treat holidays like Sundays.

historical data, or, via data from a preliminary test case run shortly prior to conducting the actual large scale survey). Running the model again, this time using 0.206 as the propensity to respond, yields the following:

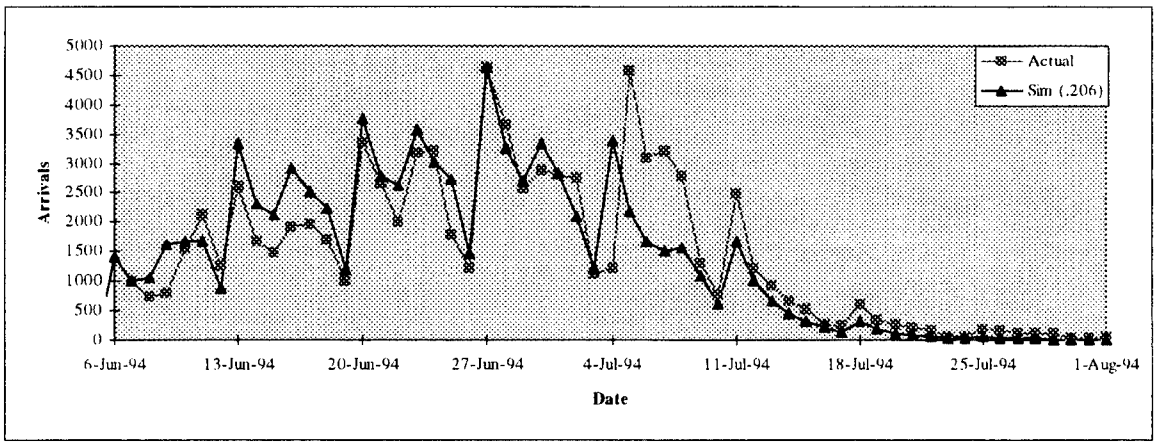

Figure 14

Here the model predicts the arrival process with surprising and obviously useful accuracy. In practice one could begin with a prior estimate of response probability, and then refine that estimate as data accumulates, and thus sequentially modify the mailing strategy on the fly. Such a dynamic control policy should be the topic of future research.

E. MAILING STRATEGIES

System limitations, time constraints, number of responses required and caller attrition levels are just a few of the factors that could impact the design of a mailing strategy. In most cases, a mailing strategy can be described in terms of goals and constraints, which suggest application of network linear/integer programming techniques. Consider the following simple deterministic model:

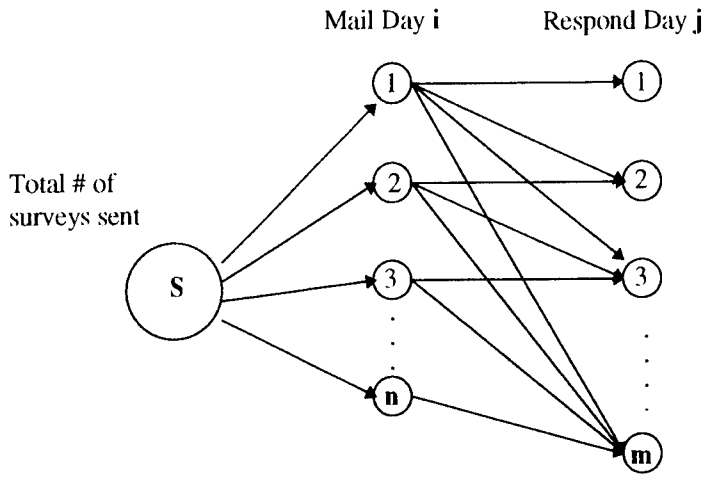

Figure 15

Node i=1 refers to the number of surveys mailed on day one and Node j=1 refers to the number of responses received on day one.

Now suppose the decision maker wishes to place a priority on early receipt of surveys (i.e. the DM needs results as soon as possible). Then a network linear programming formulation of the problem might resemble the following:

INDICES:

i = day on which surveys are mailed

j = day on which surveys are received

DATA:

S = Total number of surveys to be mailed

p_s = expected percentage of responses from a population of surveys S

c_{ij} = expected percentage of responses from a population of survey responses mailed on day i that will respond on day j

b_j = maximum number of responses to be received on day j (overload conditions as a possibility)[21]

VARIABLES:

X_i = number of responses generated from survey mailed on day i (note: X_i/p_s could be the actual number that should be mailed on day i in order to account for non-responders)

Y_j = number of responses that occur on day j

OBJECTIVE:

MINIMIZE $Z = \sum_{j=1}^{m} j * Y_j$ **(priority on surveys received early)**

CONSTRAINTS:

$$p_s * S \le \sum_{j=1}^{m} Y_j$$

$$\sum_{i=1}^{n} c_{ij} * X_i \le b_j , \quad \forall j$$

$$X_i, Y_j \ge 0 \qquad \forall i, j$$

[21] Note that if one chooses a b_j that is greater than an overload limit, the network solution might generate a mailing strategy which produces overload situations.

Using the following parameters:

S = 500,000, $p_s = 0.25$, $b_j = 7750$ $\forall j$, and c_{ij}s developed from conditional values in the model, this formulation yields a mailing pattern as follows (note: compare this strategy to the five consecutive days of 100,000 mailings used in the Model Behavior chapter):

Day	DOW	# to Mail
1	Mon	87531
2	Tue	89421
3	Wed	0
4	Thu	50386
5	Fri	0
6	Sat	0
7	Sun	0
8	Mon	54596
9	Tue	88144
10	Wed	0
11	Thu	37995
12	Fri	0
13	Sat	0
14	Sun	0
15	Mon	88026
16	Tue	3901
Total		**500000**

Table 9[22]

This pattern results in an estimated 95 % of all potential responses responding by day 23. The results of this strategy are compared with the initial strategy of five consecutive mailings (100,000 surveys each) in the first week in the following graph. Both arrival patterns represent the mean number of arrivals, for each scenario, across ten replications.

[22] Numbers are rounded to the nearest integer.

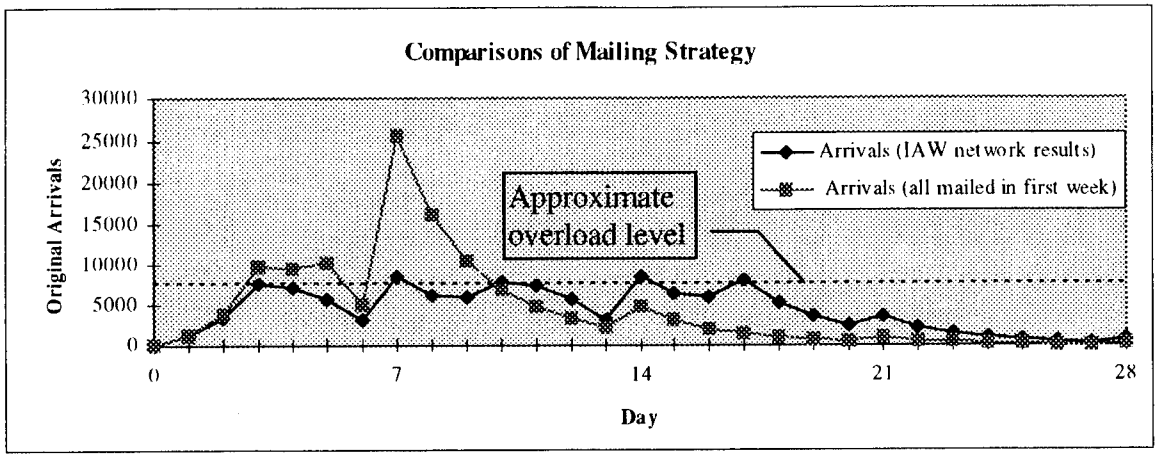

Figure 16

The network model clearly aids in systematic choice of an appropriate mailing strategy. One can see that by distributing the mailings in accordance with the recommendations of the previous table, the response to the system is more evenly distributed and, more importantly, significant overload conditions are avoided. While in the initial case 10,868 potential responses were lost due to system overload, using the network distributed model, only 146 potential responses were lost.

The network provides a better mailing strategy even with the constraint of performing all mailing within the first week (note that in the base scenario all mailings were performed in the first week). Given this additional constraint, system overload becomes inevitable. For this reason the Objective Function of the network formulation was changed to the following: MINIMIZE Z subject to the additional mailing constraint, and the constraint that $Z \geq b_j$, $\forall j$ (note that this modification causes the network formulation to become a non-linear programming problem). The modified network solution is as follows:

Day	DOW	# to Mail
1	Mon	196366
2	Tue	173996
3	Wed	58836
4	Thu	70802

Table 10[23]

While significant overload still occurs using this strategy (because of the forced concentration of surveys over a short period of time), only 7,928 potential responses are lost, 27% less than in the base case.

The network structure is easily modified to handle numerous different objectives and constraints allowing for the specific requirements of the decision maker. Additionally, as the model has been designed to encompass the possibility of multiple surveys (each with unique parameters), the network structure can be similarly expanded to:

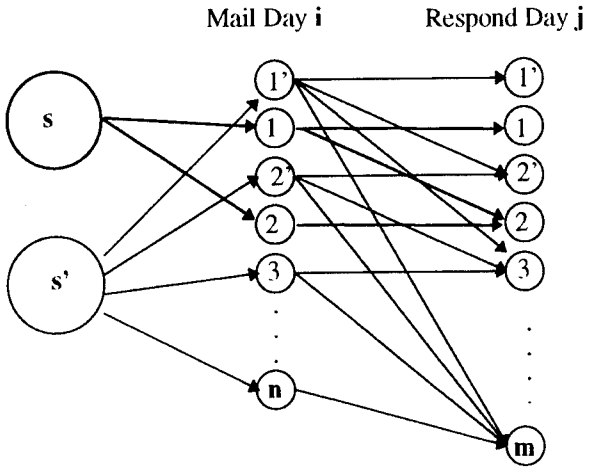

Figure 17

[23] See previous Footnote.

One problem presented by the multiple survey setup involves determining overload limits (b_j). One solution to this problem involves assigning weights to each arrival on receipt day j on the basis of the survey type of the arrival. Steps for determining weights are as follows:

- Determine the overload limit[24] for each survey type by running the model in a single survey posture and assign these values to $\theta_1 ... \theta_k$ (k survey types).
- Choose one of the limits to be the baseline limit θ_{base} and determine a multiplier h_i ($h_1 ... h_k$) for each of the limits such that $\theta_{base} = h_i * \theta_i$.
- b_j then, must be greater than or equal to the sum of all arrivals on day j weighted by the arrival's respective multiplier h_i.

A simpler approach might be to use the single survey model, fixing the proportion/mixture of the individual survey types to be mailed on a given day and assign receipt day upper bounds based on predicted overload levels for the proportional mailings. In most cases, these problems are small enough to be solved using commercial spreadsheets (as was the sample problem).[25]

---

[24] See Service Time Sensitivity - Section C.

[25] Solutions presented in this study were obtained using Microsoft Excel™ (linear/non-linear program) Solver.

VI. SIMULATION MODEL

The simulation model developed for the CATI system was written and compiled using MODSIM II (ver 1.8). MODSIM is a high-level Object-Oriented Programming language principally geared towards process simulation. The model's objects were developed in an attempt to capture effects of specific components in the Survey Process. The key objects in the simulation consist of the Manager, the SurveyGenerator, Surveys, Servers, and Customers.[26]

The remainder of the chapter describes the way in which MODSIM handles the current model.

A. THE MANAGER

The manager object simulates the functionality of the system manager. The system manager can be seen as a composite of CATI program logic and manual intervention by a system supervisor to allocate servers to call-back status. The manager's primary job involves command and control of the servers in the system. When the manager is initialized, it requires user input to define the number of servers the system will use. To keep track of the servers and their respective status, the manager possesses four basic FIFO queues,

The IdleServers Queue contains all servers tasked with conducting surveys which are currently not in service with a customer. The BusyServers queue (as the name suggests) contains all servers tasked with conducting surveys which are currently serving a customer. The CBIdleServers Servers queue contains all servers tasked with requesting

[26] Much of the inherent structure of this simulation was taken from code developed in OA3302-System Simulation with the assistance of Prof. M. Bailey (Naval Postgraduate School).

customers to call back at a later time which are currently not in service with a customer. The "busy" counter part to CBIdleServers is called CBBusyServers.

These Queues are themselves objects which are derived from an object class inherent to the MODSIM language called StatQueueObj which possesses all the functionality of a FIFO Queue plus basic statistics collection on the number of Objects in the Queue.

In MODSIM, an objects functionality is defined by its Methods (analogous to procedures). The two primary methods employed by the Manager Object are: Admit Customer and ServerReady.

1. Admit Customer Method

The Admit Customer method is called when a customer attempts to call the system. The manager checks the available server status. If there are servers in the IdleServers Queue then a server is removed from the IdleServers Queue, assigned that customer and placed in the Busy Servers queue. If there are no available 'Survey' Servers but there are available 'Call-Back' Servers (i.e. there are servers in CBIdleServers) then a server is removed from the CBIdleServers queue, assigned that customer and placed in the Call Back busy servers queue. If no servers are available, the customer receives a busy signal.

2. ServerReady Method

While Admit Customer deals with the system upon customer entry (i.e. call receipt) the ServerReady method controls the system upon customer departure. When a server completes service (regardless of whether it has been conducting a survey or requesting a call back) it asks the manager to invoke the server ready method. As in the case of Admit Customer, the manager first checks the system state in regards to server availability. If all servers are busy (both normal and call back) the manager will allocate three additional servers from normal status to call-back status upon completion of service

to their current customer (this is the DHRSC procedure). If all servers are not busy or if all servers are busy but the manager is already in the process of reallocating servers to call-back status the manager will not change the server allocation. In either case, the server is added to the appropriate Idle Queue and awaits further tasking from the manager.

B. SERVERS

Servers are analogous to the individual phone lines (although technically 24 phone lines are controlled by one of the four slave computers, the multi-tasking capability of these servers make each phone line independent, unless a computer fails. Servers possess two Boolean state variables. The first (called Busy) defines whether the server is currently busy, the second (called CB) defines whether the server is assigned to conduct a survey, or to ask callers to call back at a later time.

The server's primary methods are Serve Customer and CBServeCustomer. As these names imply, these methods cause the server to serve the customer, where service requires a variable amount of time based on either a Gamma Distribution or Exponential Distribution in case the server is assigned to survey duties or a short fixed amount of time(nominally 30 sec).[27] In the Survey case, upon completion of service the customer leaves the system completely.

C. CUSTOMERS

The customers are the individual respondents. While their arrival and departure is largely controlled by the Survey Object, they must individually decide, when not served immediately, if and when to call back. This decision logic is handled by the CheckForBalk Method. When called, this method first conducts a Bernoulli trial to determine whether

[27] 30 seconds is the approximate amount of time that Call Back server will expend relaying the 'call back message'.

the customer will attempt another call. If a call back will be initiated the customer waits an exponentially distributed amount of time and then arrives at the system again.[28]

D. SURVEYS

Survey Objects are the heart of the simulation. A survey object generates the arrival process for recipients of questionnaires mailed on a specific day. Its primary methods are GetSize, CalcArr, and GenerateCustomers.

1. GetSize Method

The GetSize Method takes the number of questionnaires mailed, **n**, (on the day for which the Survey Object represents) as input, and samples from a Binomial distribution $\sim(\mathbf{n}, \hat{\mathbf{p}})$ to determine the total number of arrivals that will occur as a result of that mailing.

2. CalcArr Method

The CalcArr Method takes the total number of arrivals (from GetSize) as input to compute daily arrivals out to the day on which the survey ceases to be serviced by the CATI system.

CalcArr's first action is to determine the residual response probabilities for all days from day of mailing to the last day of system service. Depending on which day of the week the surveys were mailed, week-zero conditional probabilities are assigned. The conditional probabilities for the following weeks are computed based on a weekly linear decline with **d**=0.3 on Monday and **d**=0.12 on Sunday (i.e., all following weeks follow the pattern described in Table 4). Once the $\mathbf{d}_{w,(j-i)}$'s have been computed, daily arrivals are drawn by sampling from a Binomial ~ (**R-Q**, $\mathbf{d}_{w,(j-i)}$) where **R-Q** is the number of customers who will call but have not yet called into the system by day j.

---

[28] This method requires specification of p for the Bernoulli Trial and μ for the exponential waiting time. There is, as yet, no data to support estimates for these parameters.

3. GenerateCustomers Method

The GenerateCustomers Method takes the daily arrivals (from CalcArr) and breaks that number into $\mathbf{c}_i$ hourly arrivals using Multinomial sampling. Once hourly arrivals are known, arrival times for individual customers are derived from $\mathbf{c}_i$ ordered samples from a Uniform(0,1) distribution. At each designated time, a Customer object is generated. Upon generation of a customer, GenerateCustomers conducts a Bernoulli trial with $\mathbf{p}$=0.835 (probability of call resulting in success) to determine whether the call will result in a completed survey. If the trial result is success, the customer will be assigned a service time based on a sample from an empirical density of successful calls. Otherwise, the customer will be assigned a service time based on a sample time from an empirical density of unsuccessful calls. The customer is then sent to the Manager. The customer retains its assigned service time (complete/incomplete) regardless of whether he/she enters a retry status.

E. THE SURVEY GENERATOR

The SurveyGenerator creates surveys and provides structure for their interaction with the simulation.[29] Its initialization Method (ObjInit) queries the user for the number of surveys, days and size of mailings for each survey, mean completion time for each type of survey, Alpha value for the Gamma distribution, minimum completion time (i.e. amount of shift in the Gamma distribution) and the day on which the survey will be removed from the system. The SurveyGenerator uses this information to generate the individual surveys.

---

[29] In this context the word survey refers the process surrounding the total mailings of a specific type of survey. As noted in the introduction, this model has been designed to include the possibility of handle multiple survey types because the system will eventually possess this capability.

1. GenerateSurveys Method

The GenerateSurveys Method creates a Survey Object for each day of mailing for a particular Survey type. For instance; if survey A has mailings scheduled for days one, three and five, then the Survey Generator will create three separate surveys. This is based on the assumption that responses to a survey from a particular mailing day are independent of responses from a different mailing day.

VII.VARIABILITY OF THE MODEL RESULTS

Because of the reliance on the Binomial and Multinomial Distributions for description of the Arrival process, the variability of model results across replications is relatively small. The model was run for a total of 30 replications with the following parameters:

- 96 servers (utilizing the default call-back server strategy)
- 125000 surveys mailed on day one of survey service (Monday)

The following chart displays the number of arrivals for the first 20 days of a simulation run across 30 replications.

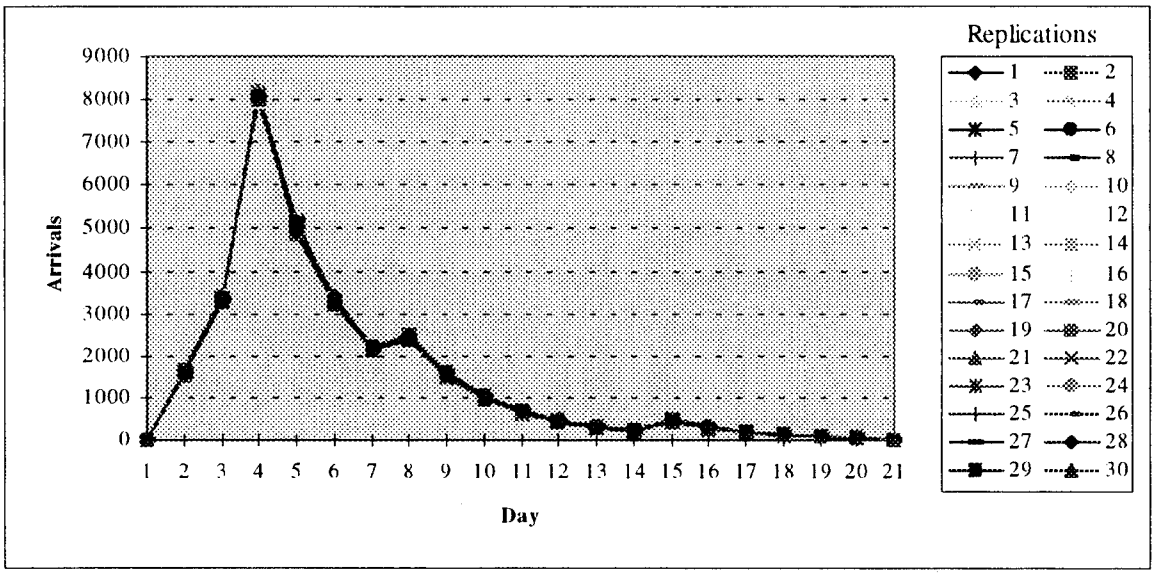

Figure 18

Taken in context of total arrivals, the variability seems almost non-existent. To demonstrate the low variation on a more appropriate scale, the following graph uses the mean number of arrivals across replications as a baseline and plots deviations from the baseline results as a ratio of the amount of deviation to the mean number of arrivals.

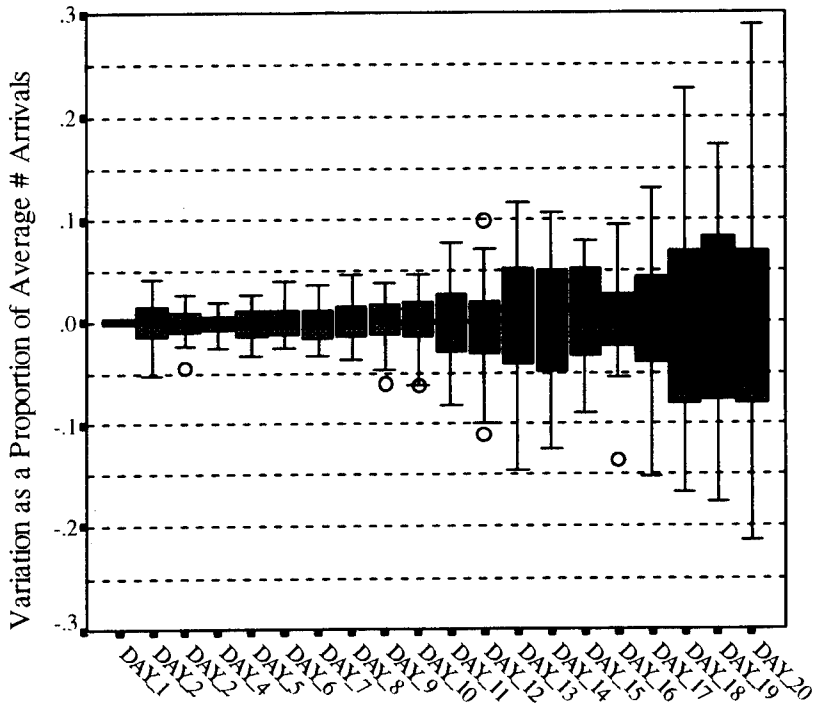

Figure 19

Clearly, the variation in arrivals across replications has minimal significance in terms of using the model as a predictor of system state. Having made this observation, it is important to note that with only one trial (i.e. the first survey mailing) it is extremely difficult to obtain an accurate description of the nature of the variation in the arrival process.

VIII.RECOMMENDATIONS

A. MODEL IMPROVEMENT

•This thesis provides an initial descriptive and predictive model for the CATI to be used as a decision aid. Given the limited data available and unusual behavior of the arrival process, simulation was chosen as the first line of attack. But simulation is time-consuming and (in many ways) not as convenient and powerful as a solid mathematical model. For this reason it seems that, combined with better and more enveloping data collection, the next line of attack should lie in the form a mathematical model.[30] Such a model should be stochastic and adaptive to evolving future conditions.

•While the current model appears to capture the behavior of the arrival process for mailed surveys, it is difficult to ascertain its applicability to surveys in general, since existing data involves only two distinct survey periods (both of the same survey type).[31] Many more survey trials will be required to validate the model in the general case, or perhaps allow for model variations depending on specific classification of the types of surveys to be serviced. In other words: as more data is obtained from future surveys, it should be possible to develop a database of appropriate parameters, applicable to a wide variety of survey types, based on historical performance of surveys with similar properties (i.e. common target population, common survey time periods, common survey service times, etc...).

•Care must be taken when assuming propensity to respond because of the wide variety of influencing factors. One option is to mail a trial set of surveys immediately prior (2-4

[30] See Gaver and Jacobs (1994) for discussion of possible mathematical models.

[31] Minor modifications were made to the survey between periods. However it is unclear whether these changes had an impact on propensity to respond, survey completion, or service times.

weeks) to large scale survey mailings to allow for more efficient planning of the mailing strategy, based on data analysis of the trial set. Properly implemented, the trial set could be easily run concurrently with other actual surveys (i.e. you do not have to leave the system free for exclusive use of the trial set).

•There are many facets to the arrival process that are unknown because of the absence proper data. One difficulty arises from the fact that arrival times are based on the difference between time of mailing and time of receipt. Clearly, some of the unexplained variability of empirical results is directly related to the time it takes for the survey to be delivered through the mailing system, the success of the postal service in correctly delivering the survey (i.e. getting it to the right address) and the quality of the database from which the mailing addresses are drawn (incorrect addresses are probably more likely in a population that frequently moves - i.e. surveys directed towards Military personnel, recent high school graduates etc...).[32] Another difficulty is that currently it is unclear who has called back (as a result of a busy signal or a call-back message), the number of times they attempted a call-back, and the underlying distribution of the wait times in a call-back scenario (making it difficult to estimate parameters for the customer call-back logic as implemented by the model). For these reasons, it is strongly recommended that an attempt be made to collect and analyze such data. A straightforward approach would be to add additional questions requesting such information as when was the survey received, was a Retry attempted and why, and how many times was a retry attempted before successfully accessing the system. However, the respondent should not be burdened with too many such questions.

•Finally, this model has been developed to be used, and to grow. Its Object Oriented Design makes significant changes fairly simple. It can already handle several types of

[32] See previous recommendation.

surveys running simultaneously. Future modifications might include features such as modeling system failure, or effects of scheduled maintenance and holidays, as well as improving the efficiency of the code and data structures.

B. EMPLOYMENT OF THE MODEL AND CATI

The primary purpose for this study was to develop a useful decision aid for the managers of the CATI system. Although, being the initial study, the current simulation model is relatively unrefined, with proper employment it can still aid in the strategic use of this valuable and powerful system.

•Using "What if?" scenarios and surface response methodology should make it possible to obtain approximate overload limits given various mixtures of survey types (each with their own behavior patterns). As touched on, in the Results and Validation Chapter, these overload limits not only will give insight into system capacity, but are key in the development of an appropriate mailing strategy.

•While there are many possible approaches to developing an effective mailing strategy (trial and error with multiple simulation runs being an inefficient example), use of variations of the network optimization models described previously, should provide a good subset of strategies to test in the model (thus reducing the number of runs required to locate an acceptable plan).

•As a predictive tool, this model could be used to determine system state at a given point in time allowing for the scheduling of maintenance during low-response periods.

•Additionally, given that it becomes possible to quantify parameters for a customer's "Call-Back logic", this model should give some indication of the magnitude of response attrition in the case of overload conditions, system failure, or immediate maintenance

requirements (information that would otherwise be undetectable because the callers would not be getting through to the system).

•As previous results show, the use of "Call Back Servers" is inefficient in a broad range of scenarios. Their use should be predicated on the ability to insure high call-back probability combined with a high degree of control over the scheduling of a retry. If retries are not pushed far enough into the future (i.e. past peak/overload periods), the Call Back Servers merely expand the amount of time the system is in overload. In general, **not using** the "Call Back Servers" seems to be the best strategy, but this is subject to further tests.

C. OTHER APPROACHES TOWARD OPTIMALITY

While mailing strategies are one aspect of controlling system demand, there are several other ways to approach this problem. Even on days where the demand on the system uses all resources for the hours of peak response, for much of the day the system is being under-utilized (in other words not very many people try to call before 5AM). Further study should be conducted in an attempt to more evenly distribute the flow of arrivals throughout the day. Ideally, one would like a uniform arrival rate 24 hours a day.

•One possible approach lies in controlling the geographic distributions of the batch mailings. For instance, if one assumes that, in general, a population is most likely to respond between the hours of 1300 and 1500 (based on individual local times) some spread could be achieved by mailing uniformly across time zones (e.g. while east coast response rates are starting to decline, the west coast response rates are starting to peak).

•Another possible approach, would be to examine the effectiveness of assigning individual respondents specific windows of time in which they are "allowed to call".

D. RELATED APPLICATIONS

- This type of model should be easily adaptable to many other processes involving transitory queues, especially in other areas of communication. Some applications are as follows:
- Resource allocation problem in satellite communications (e.g. Narrow band circuit assignment for BattleGroup/Battlefield Communications).
- Message receipt and processing (e.g. broadband SATCOM)
- Emergency Room (Battlefield MASH) unit processing capability and operating level. Here attrition takes on a much more serious note, as attrition, due to overload, would be measured in human lives.

APPENDIX - SIMULATION CODE

(1) MAIN MODULE CATI;

```
FROM RunReplications IMPORT RunReplications;
BEGIN
RunReplications;
END MODULE.
```

(2) DEFINITION MODULE WriteLineQt;

{Modification of Bailey's WriteLine-procedures for file output}

```
PROCEDURE WriteLineQt(IN n:INTEGER;IN String : STRING); PROCEDURE
WriteLineCloseQt;
END MODULE.
```

(3) IMPLEMENTATION MODULE WriteLineQt;

{Modification of Bailey's WriteLine-procedures for file output}

```
FROM IOMod IMPORT FileUseType(Output);
FROM IOMod IMPORT StreamObj;
FROM UtilMod IMPORT DateTime;
FROM SimMod IMPORT SimTime;
VAR
DT : STRING;
TraceStream : StreamObj;
PROCEDURE WriteLineQt(IN n:INTEGER;IN String : STRING);
BEGIN
IF (TraceStream = NILOBJ)
NEW(TraceStream);
ASK TraceStream TO Open("Qtim"+INTTOSTR(n)+".out", Output); DateTime(DT);
ASK TraceStream TO WriteString(DT);
ASK TraceStream TO WriteLn;
ASK TraceStream TO WriteLn;
END IF;
ASK TraceStream TO WriteReal(SimTime(), 7,3); ASK TraceStream TO WriteString(","
+ String); ASK TraceStream TO WriteLn;
END PROCEDURE;
PROCEDURE WriteLineCloseQt;
BEGIN
ASK TraceStream TO Close; DISPOSE(TraceStream);
END PROCEDURE;
END MODULE.
```

(4) DEFINITION MODULE SurveyGen;

{the survey generator object calls for input from the user defining the number and parameters for the surveys. It then generates a survey object for each batch mailing of each survey to be conducted}

```
FROM GrpMod IMPORT QueueObj;
FROM Survey IMPORT SurveyObj;
FROM RandMod IMPORT RandomObj;
TYPE
surveyfieldsType = (RespondP,ServiceTime,Alpha,MinServiceTime, {parameters}
StartingTime,DaysOfMailing, DailyMailingSize,StoppingTime);
surveyarrayType = ARRAY surveyfieldsType, INTEGER OF REAL;
sizerecType = RECORD
                    mailday : REAL;
                    mailsize: REAL;
              END RECORD;
sizearrayType = ARRAY INTEGER OF ARRAY INTEGER OF sizerecType;
SurveyQueueObj = OBJECT(QueueObj[ANYOBJ : SurveyObj]);
END OBJECT;
ALLSurveyQueueObj = OBJECT(QueueObj[ANYOBJ : SurveyQueueObj]);
END OBJECT;
SurveyGenObj = OBJECT
      n : INTEGER;    {# of surveys}
      DOW: INTEGER;   {day of week quarter starts}
      FinalTime: REAL;
      surveydata : surveyarrayType;
Sizedata : sizearrayType;
ALLSurveyQueue : ALLSurveyQueueObj; R: RandomObj;
ASK METHOD ObjInit;
ASK METHOD GetNewSize(IN N:REAL);
ASK METHOD Reset;
TELL METHOD GenerateSurveys;
END OBJECT;
VAR
SurveyGenerator : SurveyGenObj;
END MODULE.
```

(5) IMPLEMENTATION MODULE SurveyGen;

{Implementation for the Survey Generator Object}

```
FROM Survey IMPORT SurveyObj;
FROM SampleOutput IMPORT SampleOutputer;
FROM RandMod IMPORT RandomObj;
OBJECT SurveyGenObj;
{.....}
ASK METHOD ObjInit;
{.....}
VAR i,j : INTEGER;
      x:REAL;
BEGIN
NEW(R);
FinalTime:=0.0;
NEW(ALLSurveyQueue);
OUTPUT("How many surveys will you be planning for this quarter.....?"); INPUT(n);
OUTPUT("What day of week is the first day of the quarter; i.e. input 1 for Sunday");
OUTPUT("2 for Monday ... 7 for Saturday");
INPUT(DOW);
NEW(surveydata,RespondP..StoppingTime,1..n);
NEW(Sizedata,1..20,1..n);              {20 is max # of maildays}
FOR i:= 1 TO 20
      FOR j:= 1 TO n
                  NEW(Sizedata[i][j]);
      END FOR;
END FOR;
FOR i:= 1 TO n
OUTPUT("For Survey number " + INTTOSTR(i) + " please answer the following
questions"); OUTPUT("What is the individual propensity to respond?");
INPUT(surveydata[RespondP,i]);
OUTPUT("What is the average survey completion time for a single respondent (in
Minutes)?");
INPUT(surveydata[ServiceTime,i]);
OUTPUT("What is the Alpha value for survey completion time for a single respondent?");
INPUT(surveydata[Alpha,i]);
OUTPUT("What is the minimum survey completion time for a single respondent (in
Minutes)?");
INPUT(surveydata[MinServiceTime,i]);
OUTPUT("What is the survey start time (First day of quarter = 0, second = 1)?");
INPUT(surveydata[StartingTime,i]);
OUTPUT("What is the number of Mailing days?");
INPUT(surveydata[DaysOfMailing,i]);
  FOR j := 1 TO TRUNC(surveydata[DaysOfMailing,i])
    OUTPUT("What is the day of mailing day "+ INTTOSTR(j)+" (First day of quarter =
0, second = 1)?");
```

```
    INPUT(x);
    Sizedata[j,i].mailday:=x;
    OUTPUT("What is the mailing size?");
    INPUT(x);
   Sizedata[j][i].mailsize:=x;
  END FOR;
OUTPUT("What is the end day of survey service (i.e. when will it be taken off the
system)?");
INPUT(surveydata[StoppingTime,i]);
        IF surveydata[StoppingTime,i] > FinalTime
        FinalTime:=surveydata[StoppingTime,i];
        END IF;
END FOR;
END METHOD;
{.....}
ASK METHOD GetNewSize(IN N:REAL);
{.....}
 BEGIN
Sizedata[1][1].mailsize:=N;
END METHOD;
{.....}
ASK METHOD Reset;
{.....}
 VAR Survey:SurveyObj;
SurveyQueue:SurveyQueueObj;
BEGIN
 WHILE (ASK ALLSurveyQueue numberIn > 0)
                              SurveyQueue := ASK ALLSurveyQueue TO Remove();
                              WHILE (ASK SurveyQueue numberIn > 0)
                                      Survey := ASK SurveyQueue TO Remove();
                                      DISPOSE(Survey);
                              END WHILE;
                              DISPOSE(SurveyQueue);
END WHILE;
END METHOD;
{.....}
TELL METHOD GenerateSurveys;
{.....}
VAR
i,j: INTEGER;
Survey : SurveyObj;
SurveyQueue : SurveyQueueObj;
BEGIN
```

```
FOR i:= 1 TO n
        NEW(SurveyQueue);
 FOR j:= 1 TO TRUNC(surveydata[DaysOfMailing,i])
                NEW(Survey);
                ASK Survey TO GetName(INTTOSTR(i) + "," + INTTOSTR(j));
                ASK Survey TO
GetServiceParams(surveydata[ServiceTime,i]/(24.0*60.0),surveydata[Alpha,i],surveydata
[MinServiceTime,i]/(24.0*60.0));
                ASK Survey TO GetStartingTime(Sizedata[j][i].mailday);
                ASK Survey TO GetDaysOfMailing(surveydata[DaysOfMailing,i]);
                ASK Survey TO GetStoppingTime(surveydata[StoppingTime,i]);
                ASK Survey TO GetSize(Sizedata[j][i].mailsize,surveydata[RespondP,i]);
                ASK Survey TO CalcArr(FLOAT(ASK Survey Size));
                TELL Survey TO GenerateCustomers;
                ASK SurveyQueue TO Add(Survey);
 END FOR;
        ASK ALLSurveyQueue TO Add(SurveyQueue);
 END FOR;
 TELL SampleOutputer TO SampleOutput;
END METHOD;
END OBJECT;
END MODULE.
```

(6) DEFINITION MODULE Survey;

{survey objects control the arrival process for a single batch mailing of a given survey.}

```
FROM RandMod IMPORT RandomObj;
FROM ListMod IMPORT RankedList;
FROM GrpMod IMPORT QueueObj;
TYPE
Arrrec = RECORD
            n:REAL;
         END RECORD;
DayListObj = OBJECT(RankedList[ANYREC : Arrrec])    {defines ranking for
                                                      customer arrivals}
        OVERRIDE
ASK METHOD Rank(IN object1:Arrrec;IN object2:Arrrec):INTEGER;
END OBJECT;
DayListQueueObj = OBJECT(QueueObj[ANYOBJ : DayListObj])
END OBJECT;
arrivedayarrayType = ARRAY INTEGER OF INTEGER;{holds # arriving on given day}
arrivehourType = ARRAY INTEGER OF INTEGER;   {holds # arriving in given hour}
SurveyObj = OBJECT
            Name : STRING;
            Alpha: REAL;                    {                    }
            ServiceTime: REAL;              {  Gamma Params      }
            Shift: REAL;                    {                    }
            StartingTime: REAL;
            DaysOfMailing : REAL;
            DailyMailingSize: REAL;
            StoppingTime: REAL;
            Size : INTEGER;
            Qt : REAL; {number in survey who have not yet called in}
            ArriveDay:arrivedayarrayType;
            HrArr : arrivehourType;
            ASK METHOD ObjInit;
            ASK METHOD GetName(IN N:STRING);
            ASK METHOD GetServiceParams(IN N,O,P:REAL);
            ASK METHOD GetStartingTime(IN N:REAL);
            ASK METHOD GetDaysOfMailing(IN N:REAL);
            ASK METHOD GetDailyMailingSize(IN N:REAL);
            ASK METHOD GetStoppingTime(IN N:REAL);
            ASK METHOD Binomial(IN N:INTEGER;IN P:REAL):INTEGER;
            ASK METHOD Multinomial(IN N:INTEGER);
            ASK METHOD GetSize(IN N,P:REAL);
```

```
                    {calculates # of potential respondents}
                ASK METHOD GetQt(IN N:REAL);
                ASK METHOD GetArriveDay(IN I:INTEGER; IN N:INTEGER);
                ASK METHOD CalcArr(IN N:REAL);
                    {calculates arrival process}
                TELL METHOD GenerateCustomers;
                    {causes customers to arrive}
END OBJECT;
END MODULE.
```

(7) IMPLEMENTATION MODULE Survey;

```
{implementation for Survey object}
FROM Customer IMPORT CustomerObj;
FROM Manager IMPORT Manager;
FROM SurveyGen IMPORT SurveyGenerator; FROM WriteLine IMPORT WriteLine;
FROM RandMod IMPORT RandomObj;
FROM SimMod IMPORT SimTime;
FROM ListMod IMPORT RankedList;
FROM MathMod IMPORT POWER;
OBJECT DayListObj;
ASK METHOD Rank(IN rec1:Arrrec;IN rec2:Arrrec):INTEGER;
        BEGIN
                IF rec1.n > rec2.n
                        RETURN 1;
                ELSIF rec1.n < rec2.n
                        RETURN -1;
                ELSE
                        RETURN 0;
                END IF;
        END METHOD;
END OBJECT;
OBJECT SurveyObj;
{.....}
ASK METHOD ObjInit;
{.....}
BEGIN
NEW(ArriveDay,0..100);                  {100 is max number of arrival days from
                                         given batch mailing}
NEW(HrArr,1..24);
END METHOD;
{.....}
```

```
ASK METHOD GetName(IN N:STRING);
{.....}
BEGIN
Name:=N;
END METHOD;
{.....}
ASK METHOD GetServiceParams(IN N,O,P:REAL);
{.....}
BEGIN
Alpha:=O;
ServiceTime:=N;
Shift:=P;
END METHOD;
{.....}
ASK METHOD GetStartingTime(IN N:REAL);
{.....}
BEGIN
StartingTime:=N;
END METHOD;
{.....}
ASK METHOD GetDaysOfMailing(IN N:REAL);
{.....}
BEGIN
DaysOfMailing:=N;
END METHOD;
{.....}
ASK METHOD GetDailyMailingSize(IN N:REAL);
{.....}
BEGIN
DailyMailingSize:=N;
END METHOD;
{.....}
ASK METHOD GetStoppingTime(IN N:REAL);
{.....}
BEGIN
StoppingTime:=N;
END METHOD;
{.....}
ASK METHOD Binomial(IN N:INTEGER;IN P:REAL):INTEGER;
{.....}
VAR n:REAL;
    x,i:INTEGER;
BEGIN
```

```
        x:=0;
        FOR i:= 1 TO N
                n:=ASK SurveyGenerator.R TO UniformReal(0.0,1.0);
                IF n <= P
                        x:=x + 1;
                END IF;
 END FOR;
 RETURN x;
END METHOD;
{.....}
ASK METHOD Multinomial(IN N:INTEGER);
{.....}
TYPE ProbArrayType = ARRAY INTEGER OF REAL;
VAR n:REAL;
    x,i:INTEGER;
     p : ProbArrayType;
BEGIN
        FOR i:= 1 TO 24
                HrArr[i]:=0;
        END FOR;
        NEW(p,1..24);
        p[1]:=.001663;                          {CDF for multinomial draw}
        p[2]:=.002831;
        p[3]:=.004351;
        p[4]:=.008261;
        p[5]:=.019050;
        p[6]:=.043474;
        p[7]:=.085197;
        p[8]:=.140635;
        p[9]:=.204343;
        p[10]:=.271497;
        p[11]:=.345082;
        p[12]:=.425880;
        p[13]:=.515211;
        p[14]:=.608877;
        p[15]:=.695681;
        p[16]:=.768730;
        p[17]:=.827943;
        p[18]:=.880598;
        p[19]:=.923065;
        p[20]:=.955142;
        p[21]:=.976719;
        p[22]:=.989403;
```

```
        p[23]:=.996425;
        p[24]:=1.0;
        x:=0;
        FOR i:= 1 TO N
n:=ASK SurveyGenerator.R TO UniformReal(0.0,1.0); FOR x:=1 TO 24
                                IF n < p[x]
                                        HrArr[x]:=HrArr[x] + 1;
                                        EXIT;
                                END IF;
                        END FOR;
 END FOR;
END METHOD;
{.....}
ASK METHOD GetSize(IN N,P:REAL);
{.....}
BEGIN
 Size:=ASK SELF TO Binomial(TRUNC(N),P);
END METHOD;
{.....}
ASK METHOD GetQt(IN N:REAL);
{.....}
BEGIN
Qt:=N;
END METHOD;
{.....}
ASK METHOD GetArriveDay(IN I:INTEGER;IN N:INTEGER);
{.....}
BEGIN
ArriveDay[I]:=N;
END METHOD;
{.....}
ASK METHOD CalcArr(IN N:REAL);
{.....}
TYPE ProbArrayType = ARRAY INTEGER OF REAL;
VAR Day,i,j,n,Resid:INTEGER;
                    p:REAL;
                Prob:ProbArrayType;
BEGIN
Day:=(ASK SurveyGenerator DOW + TRUNC(StartingTime)) MOD 7;
n:=MAXOF(TRUNC(StoppingTime-StartingTime)-1,7); {prevent crash if length is less
                                                        than 7 days}

{establishes conditional probabilities for arrivals based on day-of-week of mailing
```

```
day 1 = Sunday, 2 = Monday ... 7 = Saturday -- assumes no mailings on Saturday or
Sunday}
CASE Day
            WHEN 2  : NEW(Prob,0..n);
                      Prob[1]:=.05;
                      Prob[2]:=.11;
                      i:=Day+3;
                      j:=3;                          {now its Thursday}
                      WHILE j <= (n)
                            p:=.3;          {peak}
                            WHILE i <= 8      {till the next week}
                                  IF j<=n
                                  Prob[j]:=p;
                                  p:=p-.03;       {.2/7  drops .2 in a week}
                                  END IF;
                            j:=j+1;
                            i:=i+1;
                            END WHILE;
                      i:=Day;
                      END WHILE;
      Resid:=Size;
      FOR i:= 0 TO n
      ASK SELF TO GetArriveDay(i,ASK SELF TO Binomial(Resid,Prob[i]));
      Resid:=Resid-ArriveDay[i];
      END FOR;
WHEN 3  : NEW(Prob,0..n);
            Prob[1]:=.05;
            Prob[2]:=.1;
            Prob[3]:=.18;
            Prob[4]:=.15;
            Prob[5]:=.09;
            i:=Day-1;                    {now its Monday}
            j:=6;
            WHILE j <= (n)
                  p:=.3;          {peak}
                  WHILE i <= 8      {till the next week}
                        IF j<=n
                        Prob[j]:=p;
                        p:=p-.03;       {.2/7  drops .2 in a week}
                        END IF;
                  j:=j+1;
                  i:=i+1;
                  END WHILE;
```

```
                    i:=Day-1;
                END WHILE;
        Resid:=Size;
        FOR i:= 0 TO n
        ASK SELF TO GetArriveDay(i,ASK SELF TO Binomial(Resid,Prob[i]));
        Resid:=Resid-ArriveDay[i];
        END FOR;
WHEN 4  :    NEW(Prob,0..n);
                    Prob[1]:=.04;
                    Prob[2]:=.05;
                    Prob[3]:=.09;
                    Prob[4]:=.04;
                    i:=Day-2;                    {now its Monday}
                    j:=5;
                    WHILE j <= (n)
                        p:=.3;          {peak}
                    WHILE i <= 8     {till the next week}
                        IF j<=n
                        Prob[j]:=p;
                        p:=p-.03;       {.2/7  drops .2 in a week}
                    END IF;
                    j:=j+1;
                    i:=i+1;
                    END WHILE;
                    i:=Day-2;
                    END WHILE;
        Resid:=Size;
        FOR i:= 0 TO n
        ASK SELF TO GetArriveDay(i,ASK SELF TO Binomial(Resid,Prob[i]));
        Resid:=Resid-ArriveDay[i];
        END FOR;
WHEN 5  : NEW(Prob,0..n);
            Prob[1]:=.02;
            Prob[2]:=.1;
            Prob[3]:=.04;
            i:=Day-3;                    {now its Monday}
            j:=4;
            WHILE j <= (n)
                p:=.3;          {peak}
            WHILE i <= 8     {till the next week}
                IF j<=n
                Prob[j]:=p;
                p:=p-.03;       {.2/7  drops .2 in a week}
```

```
                    END IF;
                    j:=j+1;
                    i:=i+1;
                END WHILE;
                i:=Day-3;
                END WHILE;
        Resid:=Size;
        FOR i:= 0 TO n
        ASK SELF TO GetArriveDay(i,ASK SELF TO Binomial(Resid,Prob[i]));
        Resid:=Resid-ArriveDay[i];
        END FOR;
WHEN 6  : NEW(Prob,0..n);
                Prob[1]:=.02;
                Prob[2]:=.01;
                i:=Day-4;                    {now its Monday}
                j:=3;
                WHILE j <= (n)
                        p:=.3;          {peak}
                WHILE i <= 8      {till the next week}
                        IF j<=n
                        Prob[j]:=p;
                        p:=p-.03;       {.2/7  drops .2 in a week}
                        END IF;
                        j:=j+1;
                        i:=i+1;
                END WHILE;
                i:=Day-4;
                END WHILE;
        Resid:=Size;
        FOR i:= 0 TO n
        ASK SELF TO GetArriveDay(i,ASK SELF TO Binomial(Resid,Prob[i]));
        Resid:=Resid-ArriveDay[i];
        END FOR;
END CASE;
END METHOD;
{.....}
TELL METHOD GenerateCustomers;
{.....}
VAR
Customer : CustomerObj;
i,j,k: INTEGER;
t,x: REAL;
n:Arrrec;
```

```
DayList: DayListObj;
DayListQueue:DayListQueueObj;
BEGIN
i:=TRUNC(StoppingTime-StartingTime);
j:=1;
NEW(DayListQueue);
WAIT DURATION StartingTime
        END WAIT;
Multinomial(ArriveDay[0]);
FOR k:=1 TO 24
                        NEW(DayList);
                        FOR i:=1 TO HrArr[k]
                                        NEW(n);
                                n.n:=ASK SurveyGenerator.R TO
                                UniformReal(0.0,1.0/24.0);
                                ASK DayList TO Add(n);
                                END FOR;
                                ASK DayListQueue TO Add(DayList);
                        END FOR;

WHILE (SimTime() < StoppingTime)
Multinomial(ArriveDay[j]); FOR k:=1 TO 24 NEW(DayList);
FOR i:=1 TO HrArr[k] NEW(n);
        n.n:=ASK SurveyGenerator.R TO UniformReal(0.0,1.0/24.0);
        ASK DayList TO Add(n);
END FOR;
ASK DayListQueue TO Add(DayList);
END FOR;
j:=j+1;
END IF;
WHILE (ASK DayListQueue numberIn > 0)
x:=0.0;
DayList := ASK DayListQueue TO Remove();
WHILE (ASK DayList numberIn > 0);
        n:= ASK DayList TO Remove();
        WAIT DURATION (n.n - x)
        END WAIT;
        x:=n.n;
        NEW(Customer);
        ASK Customer TO GetName(Name + "Customer " +  INTTOSTR(i));
        IF (ASK SELF TO Binomial(1,.835)>0);
        ASK Customer TO GetServiceMean((ASK SurveyGenerator.R TO
        Gamma(ServiceTime-Shift,Alpha)+Shift));
```

```
        ELSE
        ASK Customer TO GetServiceMean(ASK SurveyGenerator.R TO
        Exponential(.00205));{177 sec}
        END IF;
        ASK Customer TO GetSurvey(SELF);
        INC(i);
        ASK Manager TO AdmitCustomer(Customer);
        END WHILE;
        WAIT DURATION ((1.0/24.0)-x)
        END WAIT;
        END WHILE;
        IF SimTime < 1.0
        WAIT DURATION    1.0-SimTime+.000001
        END WAIT;
        END IF;
END WHILE;
END METHOD;
END OBJECT;
END MODULE.
```

(8) DEFINITION MODULE Server;

{a server object elapses time while serving a customer or telling him to call back (CB).}

```
FROM Customer IMPORT CustomerObj;
FROM RandMod IMPORT RandomObj;
TYPE
ServerObj = OBJECT
            Name : INTEGER;
            ServiceRate : REAL;
            CurrentCustomer : CustomerObj;
            Busy : BOOLEAN;
            CB : BOOLEAN;
            TELL METHOD ServeCustomer(IN Customer : CustomerObj);
            TELL METHOD CBServeCustomer(IN Customer : CustomerObj);
                  {used when in call back mode}
            ASK METHOD Reset;
            ASK METHOD GetName(IN N :INTEGER);
            ASK METHOD ChangeCB;          {changes from CB to normal}
            ASK METHOD GetServiceRate(IN N : REAL);
            ASK METHOD ObjInit;
END OBJECT;
END MODULE.
```

(9) IMPLEMENTATION MODULE Server;

{implementation code for server object}

```
FROM Customer IMPORT CustomerObj;
FROM Manager IMPORT Manager;
FROM WriteLine IMPORT WriteLine;
FROM SimMod IMPORT SimTime;
FROM RandMod IMPORT RandomObj;
OBJECT ServerObj;
{.....}
ASK METHOD ObjInit;
{.....}
BEGIN
CB:=FALSE;
END METHOD;
{.....}
ASK METHOD Reset;
{.....}
```

```
BEGIN
Busy := FALSE;
IF CurrentCustomer <> NILOBJ
        DISPOSE(CurrentCustomer);
END IF;
END METHOD;
{.....}
ASK METHOD GetName(IN N : INTEGER);
{.....}
BEGIN
Name := N;
END METHOD;
{.....}
ASK METHOD ChangeCB;
{.....}
BEGIN
IF CB
        CB:=FALSE;
ELSE
        CB:=TRUE;
END IF;
END METHOD;
{.....}
ASK METHOD GetServiceRate(IN N : REAL);
{.....}
BEGIN
ServiceRate:= N;
END METHOD;
{.....}
TELL METHOD ServeCustomer(IN Customer : CustomerObj);
{.....}
VAR
t : REAL;
R : RandomObj;
BEGIN
R := ASK Manager R;
Busy := TRUE;
CurrentCustomer := Customer;
ASK CurrentCustomer TO GetServer(SELF);
ASK CurrentCustomer.MySurvey TO GetQt((ASK CurrentCustomer.MySurvey
Qt)+1.0);
t := ASK CurrentCustomer ServiceMean;
WAIT DURATION t
```

```
ON INTERRUPT
END WAIT;
ASK CurrentCustomer.MySurvey TO GetQt((ASK CurrentCustomer.MySurvey Qt)-1.0);
Busy := FALSE;
CurrentCustomer := NILOBJ;
DISPOSE(Customer);
ASK Manager TO ServerReady(SELF);
END METHOD;
{.....}
 TELL METHOD CBServeCustomer(IN Customer : CustomerObj);
{.....}
VAR
t : REAL;
BEGIN
Busy := TRUE;
CurrentCustomer := Customer;
t := 30.0/(60.0*60.0*24.0);                    {30 second service time}
WAIT DURATION t
ON INTERRUPT
END WAIT;
ASK Manager.WaitingCustomers TO Add(Customer);
TELL Customer TO CheckForBalk(ASK Manager CallBackMeanTime,ASK Manager
CallBackProb);
Busy := FALSE;
ASK Manager TO ServerReady(SELF);
END METHOD;
END OBJECT;
END MODULE.
```

(10)DEFINITION MODULE SampleOutput;

{sample outputer controls the output files from the simulation}

```
TYPE
SampleOutputObj = OBJECT
        t:REAL;
        hi:REAL;
        low:REAL;
        n:INTEGER;
        ASK METHOD ObjInit;
        ASK METHOD INC;
        TELL METHOD SampleOutput;
        TELL METHOD GetOutput;
END OBJECT;
VAR SampleOutputer:SampleOutputObj;
END MODULE.
```

(11)IMPLEMENTATION MODULE SampleOutput;

{implementation code for sampleoutputer object - creates file output from simulation}

```
FROM WriteLineQt IMPORT WriteLineQt; FROM WriteLineQt IMPORT
WriteLineCloseQt; FROM WriteLine IMPORT WriteLine;
FROM WriteLine IMPORT WriteLineClose; FROM WriteLineCQ IMPORT
WriteLineCQ; FROM SimMod IMPORT SimTime;
FROM SurveyGen IMPORT SurveyGenerator; FROM Manager IMPORT Manager;
OBJECT SampleOutputObj;
{.....}
ASK METHOD ObjInit;
{.....}
BEGIN
OUTPUT("What day should I start collecting output?");
INPUT(low);
OUTPUT("What day should I stop collecting output?");
INPUT(hi);
OUTPUT("What is the snapshot interval?");
INPUT(t);
n:=1;
END METHOD;
{.....}
ASK METHOD INC;
{.....}
```

```
BEGIN
n:=n+1;
END METHOD;
{.....}
TELL METHOD SampleOutput;
{.....}
BEGIN
WAIT DURATION low
END WAIT;
 WHILE (SimTime() < hi)
        WAIT DURATION t
        END WAIT;
WriteLineQt(1,INTTOSTR(n)+","+INTTOSTR(ASK Manager.BusyServers
        numberIn)+","+INTTOSTR(ASK Manager.WaitingCustomers numberIn)+","
        +INTTOSTR(ASK Manager.CBBusyServers numberIn)+","+INTTOSTR(ASK
        Manager.IdleServers numberIn)+","+INTTOSTR(ASK Manager.CBIdleServers
        numberIn)); OUTPUT(REALTOSTR(SimTime()));

                ASK Manager.BusyServers TO Reset;

 END WHILE;
 END METHOD;
 {.....}
TELL METHOD GetOutput;
{.....}
BEGIN
WHILE (SimTime() < ((ASK SurveyGenerator FinalTime)+.5))
                WAIT DURATION 1.0
                END WAIT;
WriteLine(INTTOSTR(n)+","+INTTOSTR(ASK Manager
TotalArrivals)+","+INTTOSTR(ASK Manager Arrivals)+","+INTTOSTR(ASK Manager
EnterSystem)); OUTPUT(REALTOSTR(SimTime()));
        ASK Manager TO GetTotalArrivals(0);
        ASK Manager TO GetArrivals(0);
        ASK Manager TO GetEnterSystem(0);

 END WHILE;
 END METHOD;
 END OBJECT;
END MODULE.
```

(12)DEFINITION MODULE RunReplications;

{drives the simulation}

```
PROCEDURE RunReplications;
END MODULE.
```

(13)IMPLEMENTATION MODULE RunReplications;

{implementation of replication procedure}

```
FROM SampleOutput IMPORT SampleOutputer;
FROM SurveyGen IMPORT SurveyGenerator;
FROM Manager IMPORT Manager;
FROM Survey IMPORT SurveyObj;
FROM WriteLine IMPORT WriteLine;
FROM SimMod IMPORT StartSimulation, ResetSimTime; FROM RandMod IMPORT
RandomObj;
FROM WriteLineCQ IMPORT WriteLineCloseCQ;
PROCEDURE RunReplications;
TYPE
VAR
i,n:INTEGER;
R:RandomObj;
BEGIN
NEW(R);
NEW(Manager);
NEW(SurveyGenerator);
NEW(SampleOutputer);
OUTPUT("How many replications?");
INPUT(n);
i:=1;
REPEAT

OUTPUT("      R E P L I C A T I O N     " + INTTOSTR(i)); {WriteLine(" ");}
TELL SurveyGenerator TO GenerateSurveys;
TELL SampleOutputer TO SampleOutput;
TELL SampleOutputer TO GetOutput; StartSimulation;
ResetSimTime(0.0);
ASK Manager TO Reset;
ASK SampleOutputer TO INC;
i:=i+1;
UNTIL i = n+1;
```

```
END PROCEDURE;
END MODULE.
```

(14)DEFINITION MODULE Manager;

{the manager controls the CATI system, also keeps track of servers and customers in retry status via queues; keeps track of various arrival counts}

```
FROM GrpMod IMPORT StatQueueObj;
FROM Customer IMPORT CustomerObj;
FROM Server IMPORT ServerObj;
FROM RandMod IMPORT RandomObj;
TYPE
CustomerQueueObj = OBJECT(StatQueueObj[ANYOBJ : CustomerObj]) END
OBJECT;
ServerQueueObj = OBJECT(StatQueueObj[ANYOBJ : ServerObj]) END OBJECT;
ManagerObj = OBJECT
             CBn:INTEGER;
             count:INTEGER;
             maxCB:INTEGER;              {max # of servers in call back status}
             TotalServers: INTEGER;
             Arrivals:INTEGER;           {original arrivals}
             TotalArrivals:INTEGER;
             TotalInQ:INTEGER;           {total # put into retry status}
             EnterSystem:INTEGER;        {# served by server conducting surveys}
             CallBackMeanTime:REAL;      {params for retry logic in the event of}
             CallBackProb:REAL;          {getting a call back server}
             CallBackMeanTimeBusy:REAL; {params for retry logic in the event of}
             CallBackProbBusy:REAL;      {getting a busy signal}

             DECREMENTING: BOOLEAN;  {is manager in process of adding call
                                      back servers?}
             WaitingCustomers : CustomerQueueObj;
             IdleServers : ServerQueueObj;
             BusyServers : ServerQueueObj;
             CBIdleServers: ServerQueueObj;
             CBBusyServers: ServerQueueObj;
             R : RandomObj;
             ASK METHOD GetTotalArrivals(IN N:INTEGER);
             ASK METHOD GetArrivals(IN N:INTEGER);
             ASK METHOD GetEnterSystem(IN N:INTEGER);
             ASK METHOD AdmitCustomer(IN Customer : CustomerObj);
             ASK METHOD ServerReady(IN Server :ServerObj);
             ASK METHOD Reset;
             ASK METHOD ObjInit;
END OBJECT;
```

```
VAR
Manager : ManagerObj;
END MODULE.
```

(15)IMPLEMENTATION MODULE Manager;

```
{implementation code for manager object}
FROM SampleOutput IMPORT SampleOutputer;
FROM Server IMPORT ServerObj;
FROM Customer IMPORT CustomerObj;
FROM SimMod IMPORT SimTime;
FROM WriteLine IMPORT WriteLine;
FROM WriteLineCQ IMPORT WriteLineCQ;
OBJECT ManagerObj;
{.....}
ASK METHOD ObjInit;
{.....}
VAR
i :INTEGER;
n : INTEGER;
Server : ServerObj;
BEGIN
NEW(WaitingCustomers);
NEW(IdleServers);
NEW(BusyServers);
NEW(CBIdleServers);
NEW(CBBusyServers);
OUTPUT("How many servers ?");
INPUT(TotalServers);
OUTPUT("What is call back probability given caller receives a msg?");
INPUT(CallBackProb);
OUTPUT("What is call back mean time given caller receives a msg?");
INPUT(CallBackMeanTime);
OUTPUT("What is call back probability given caller receives a busy signal?");
INPUT(CallBackProbBusy);
OUTPUT("What is call back mean time given caller receives a busy signal?");
INPUT(CallBackMeanTimeBusy);
FOR i := 1 TO (TotalServers-3)
        NEW(Server);
        ASK Server TO GetName(i);
        ASK IdleServers TO Add(Server);
END FOR;
```

```
FOR i := (TotalServers-2) TO TotalServers
        NEW(Server);
        ASK Server TO GetName(i);
        ASK CBIdleServers TO Add(Server);
        ASK Server TO ChangeCB;
END FOR;
NEW(R);
DECREMENTING:=FALSE;
CBn:=0;
count:=0;
END METHOD;
{.....}
ASK METHOD GetTotalArrivals(IN N:INTEGER);
{.....}
BEGIN
TotalArrivals:=N;
END METHOD;
{.....}
ASK METHOD GetArrivals(IN N:INTEGER);
{.....}
BEGIN
Arrivals:=N;
END METHOD;
{.....}
ASK METHOD GetEnterSystem(IN N:INTEGER);
{.....}
BEGIN
EnterSystem:=N;
END METHOD;
{.....}
ASK METHOD Reset;
{.....}
VAR
Server : ServerObj;
Customer: CustomerObj;
BEGIN
WHILE (ASK BusyServers numberIn > 0)
        Server := ASK BusyServers TO Remove();
        ASK IdleServers TO Add(Server);
        ASK Server TO Reset;
END WHILE;
WHILE (ASK CBBusyServers numberIn > 0)
        Server := ASK CBBusyServers TO Remove();
```

```
                    ASK CBIdleServers TO Add(Server);
        ASK Server TO Reset;
END WHILE;
WHILE (ASK CBIdleServers numberIn > 3)
FOREACH Server IN CBIdleServers OUTPUT(INTTOSTR(Server.Name));
        IF Server.Name < (TotalServers-2)
        ASK CBIdleServers TO RemoveThis(Server);
        ASK IdleServers TO Add(Server);
        ASK Server TO ChangeCB;
        END IF;
END FOREACH;
END WHILE;
WHILE (ASK WaitingCustomers numberIn > 0)
        Customer := ASK WaitingCustomers TO Remove();
        DISPOSE(Customer);
END WHILE;
Arrivals:=0;
TotalArrivals:=0;
TotalInQ:=0;
EnterSystem:=0;
END METHOD;
{.....}
ASK METHOD AdmitCustomer(IN Customer : CustomerObj);
{.....}
VAR
Server : ServerObj;
BEGIN
IF NOT(Customer.Retrying)
        Arrivals :=Arrivals + 1;
END IF;
TotalArrivals:=TotalArrivals + 1;
IF (ASK IdleServers numberIn > 0)
        Server := ASK IdleServers TO Remove();
        ASK BusyServers TO Add(Server);
        TELL Server TO ServeCustomer(Customer);
        EnterSystem:=EnterSystem + 1;
ELSIF (ASK CBIdleServers numberIn > 0)
        Server := ASK CBIdleServers TO Remove();
        ASK CBBusyServers TO Add(Server);
        TELL Server TO CBServeCustomer(Customer);
        TotalInQ:=TotalInQ + 1;
{check for balk and custQ entry taken care of in CBServeCustomer}
ELSE
```

```
        TotalInQ:=TotalInQ + 1;
        ASK WaitingCustomers TO Add(Customer);
TELL Customer TO CheckForBalk(CallBackMeanTimeBusy,CallBackProbBusy);
END IF;
END METHOD;
{.....}
ASK METHOD ServerReady(IN Server : ServerObj);
{.....}
VAR
Server1:ServerObj;
Customer : CustomerObj;
BEGIN
IF DECREMENTING
        IF Server.CB
                ASK CBBusyServers TO RemoveThis(Server);
                ASK CBIdleServers TO Add(Server);
        ELSIF Server.Name >TotalServers-CBn
                        ASK BusyServers TO RemoveThis(Server);
                        ASK CBIdleServers TO Add(Server);
                        ASK Server TO ChangeCB;
                        count:=count + 1;
                        IF count = 3
                        DECREMENTING := FALSE; count:=0;
                        END IF;
        ELSE
                        ASK BusyServers TO RemoveThis(Server);
                        ASK IdleServers TO Add(Server);
        END IF;
ELSE
        IF ASK IdleServers numberIn = 0
                IF ASK CBIdleServers numberIn = 0
                        CBn:=ASK CBBusyServers numberIn + 3;
                        {then system is full}
                        IF CBn > maxCB
                        maxCB:=CBn;
                        END IF;
                        DECREMENTING := TRUE;
                                IF Server.CB
                                        ASK CBBusyServers TO RemoveThis(Server);
                                        ASK CBIdleServers TO Add(Server);
                                ELSIF Server.Name >    TotalServers-CBn
                                        ASK BusyServers TO RemoveThis(Server);
                                        ASK CBIdleServers TO Add(Server);
```

```
                        ASK Server TO ChangeCB;
                        count:=count+1;
                    ELSE
                        ASK BusyServers TO RemoveThis(Server);
                        ASK IdleServers TO Add(Server);
                    END IF;
            ELSE
                IF Server.CB
                    ASK CBBusyServers TO RemoveThis(Server);
                    ASK CBIdleServers TO Add(Server);
                ELSE
                    ASK BusyServers TO RemoveThis(Server);
                    ASK IdleServers TO Add(Server);
                END IF;
            END IF;
        ELSE
            IF Server.CB
                    ASK CBBusyServers TO RemoveThis(Server);
                    ASK CBIdleServers TO Add(Server);
                ELSE
                    ASK BusyServers TO RemoveThis(Server);
                    ASK IdleServers TO Add(Server);
                END IF;
            IF ASK CBIdleServers numberIn > 3
                FOREACH Server1 IN CBIdleServers
                    IF Server1.Name < (TotalServers-2)
                        ASK CBIdleServers TO RemoveThis(Server1);
                        ASK IdleServers TO Add(Server1);
                        ASK Server1 TO ChangeCB;
                    END IF;
                END FOREACH;
            END IF;
        END IF;
END IF;
END METHOD;
END OBJECT;
END MODULE.
```

(16)DEFINITION MODULE Customer;

{customer objects arrive and leave; their only big job is to decide if they should retry and if so, when}

```
FROM Survey IMPORT SurveyObj;
FROM Server IMPORT ServerObj;
TYPE
CustomerObj = OBJECT
        Name : STRING;
        Retrying : BOOLEAN;
        ServiceMean: REAL;
        EntryTime : REAL;
        MyServer : ServerObj;
        MySurvey : SurveyObj;
        ASK METHOD ObjInit;
        ASK METHOD GetServiceMean(IN N:REAL);
        ASK METHOD GetEntryTime(IN N:REAL);
        ASK METHOD GetSurvey(IN S : SurveyObj);
        ASK METHOD GetServer(IN S : ServerObj);
        ASK METHOD GetName(IN S : STRING);
        TELL METHOD CheckForBalk(IN T,P : REAL);
END OBJECT;
END MODULE.
```

(17)IMPLEMENTATION MODULE Customer;

```
{implementation for customer object}
FROM Survey IMPORT SurveyObj;
FROM Server IMPORT ServerObj;
FROM SurveyGen IMPORT SurveyGenerator;
FROM WriteLine IMPORT WriteLine;
FROM Manager IMPORT Manager;
OBJECT CustomerObj;
{.....}
ASK METHOD ObjInit;
{.....}
BEGIN
EntryTime:=0.0;
Retrying:=FALSE;
END METHOD;
{.....}
```

```
ASK METHOD GetName(IN N : STRING);
{.....}
BEGIN
Name := N;
END METHOD;
{.....}
ASK METHOD GetServiceMean(IN N : REAL);
{.....}
BEGIN
ServiceMean:= N;
END METHOD;
{.....}
ASK METHOD GetEntryTime(IN N : REAL);
{.....}
BEGIN
IF EntryTime = 0.0;
        EntryTime := N;
END IF;
END METHOD;
{.....}
ASK METHOD GetSurvey(IN S : SurveyObj);
{.....}
BEGIN
MySurvey := S;
END METHOD;
{.....}
ASK METHOD GetServer(IN S : ServerObj);
{.....}
BEGIN
MyServer := S;
END METHOD;
{.....}
TELL METHOD CheckForBalk(IN T,P : REAL); {T:mean Time till call back,P:prob}
{.....}
VAR t: REAL;
BEGIN
t:=ASK SurveyGenerator.R TO UniformReal(0.0,1.0);
IF t <= P
Retrying:=TRUE;
t:=ASK SurveyGenerator.R TO Exponential(T);
        WAIT DURATION t
        END WAIT;
ASK Manager.WaitingCustomers TO RemoveThis(SELF);
```

```
        ASK Manager TO AdmitCustomer(SELF);
 ELSE
ASK Manager.WaitingCustomers TO RemoveThis(SELF);
        DISPOSE(SELF);
 END IF;
END METHOD;
END OBJECT;
END MODULE.
```

LIST OF REFERENCES

1. Law, A.M., Kelton, W. D., *Simulation Modeling and Analysis*, 2d ed., McGraw-Hill, Inc., 1991.
2. Gaver, D.P., Jacobs, P.A., Ellis, N.R., "Transitory Queues in Telephone Response to a Survey", lecture presented at TIMS XXXII, Anchorage, Alaska, 14 June 1994.
3. Defense Health Resources Study Center, *Point Paper-Response Rate to Military Health Services System (MHSS) Health Beneficiary Survey*, by G. Iversen, 2 August 1994.

BIBLIOGRAPHY

Ahuja,R.K., Magnanti, T.L., and Orlin, J.B., *Network Flows-Theory, Algorithms, and Applications*, Prentice-Hall, Inc., 1993.

CACI Products Company, *MODSIM II-The Language for Object-Oriented Programming-Reference Manual*, 1993.

Defense Health Resources Study Center, *Point Paper-Response Rate to Military Health Services System (MHSS) Health Beneficiary Survey*, by G. Iversen, 2 August 1994.

Gaver, D.P., Jacobs, P.A., Ellis, N.R., "Transitory Queues in Telephone Response to a Survey", lecture presented at TIMS XXXII, Anchorage, Alaska, 14 June 1994.

Gaver, D.P., Lehoczky, J.P., and Perlas, M.P., "Service Systems with Transitory Demand", *Logistics, North-Holland/TIMS Studies in the Management Sciences*, v. 1, pp. 21-34, 1975.

Gonzalez, M.E., Kasprzyk, D., and Scheuren, F., "Nonresponse in Federal Surveys: An Exploratory Study", *Amstat News*, v. 208, pp.1-7, April 1994.

Law, A.M., Kelton, W. D., *Simulation Modeling and Analysis*, 2d ed., McGraw-Hill, Inc., 1991.

Mendenhall, W., Wackerly, D.D., and Scheaffer, R.L., *Mathematical Statistics with Applications*, PWS-KENT, 1990.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor D. P. Gaver OR/Gv Dept. of Operations Research Naval Postgraduate School Monterey, California 93943	5
4.	Professor P. A. Jacobs OR/Jc Dept. of Operations Research Naval Postgraduate School Monterey, California 93943	1
5.	Dr. E. Schmitz Navy Recruiting Command Code 22 4015 Wilson Boulevard Arlington, Virginia 22203-1991	1
6.	Capt. G. Iversen Defense Health Resources Study Center Naval Postgraduate School Monterey, California 93943	3
7.	LT Neale R. Ellis 8304 Epinard Court Annandale, Virginia 22003	3